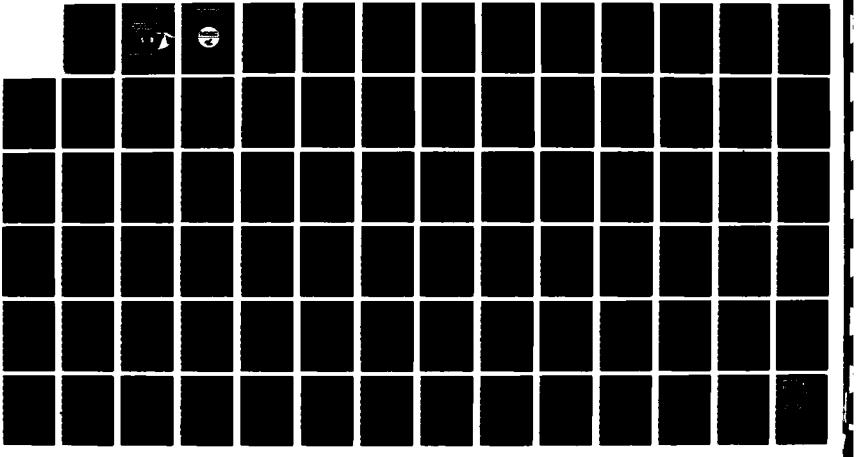
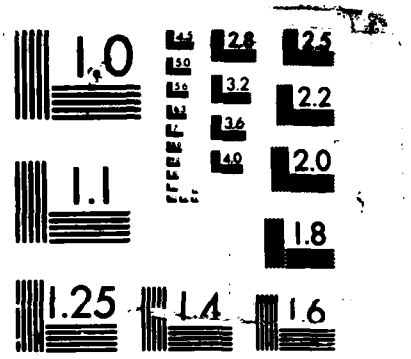
AD-A181 939 THE SEGMENTED WAVEGUIDE PROGRAM FOR LONG WAVELENGTH PROPAGATION CALCULATIONS(U) NAVAL OCEAN SYSTEMS CENTER SAN DIEGO CA J A FERGUSON ET AL APR 87 NOSC/TD-1071 1/1

UNCLASSIFIED F/G 20/14 NL





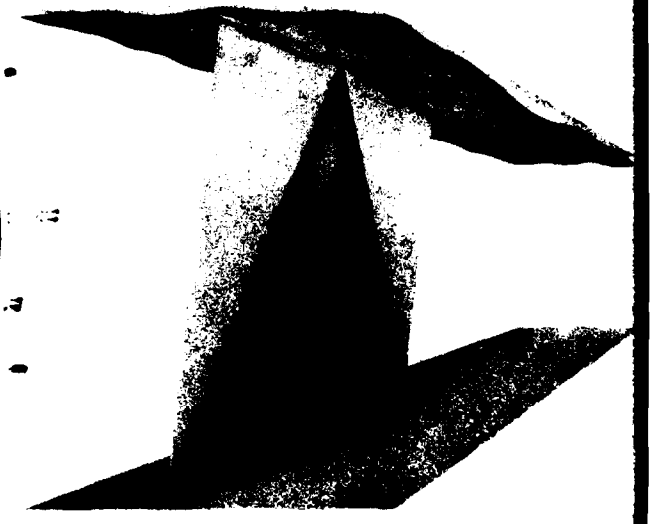
MICROCOPY RESOLUTION TEST CHART

(12)

Technical Document 1071
April 1987

The Segmented Waveguide Program for Long Wavelength Propagation Calculations

J. A. Ferguson
F. P. Snyder



Naval Ocean Systems Center
San Diego, California 92152-5000

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

E. G. SCHWEIZER, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work was performed by Code 544 of Naval Ocean Systems Center and funded by the Defense Nuclear Agency.

Released by
J.A. Ferguson, Head
Modeling Branch

Under authority of
J.H. Richter, Head
Ocean and Atmospheric
Sciences Division

JJ

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A181939

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC TD 1071	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center 6b. OFFICE SYMBOL (if applicable) NOSC	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000	7b ADDRESS (City, State and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION Defense Nuclear Agency 8b OFFICE SYMBOL (if applicable) DNA	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State and ZIP Code) Washington, DC 20305	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 62715H; PROJECT NO: MP20; TASK NO:; AGENCY ACCESSION NO: DN651 524

11 TITLE (include Security Classification)
The Segmented Waveguide Program for Long Wavelength Propagation Calculations

12 PERSONAL AUTHOR(S)
J.A. Ferguson and F.P. Snyder

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Interim	FROM Jan 1986 TO Dec 1986	April 1987	79

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Longwave propagation wavelength mode propagation path

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A computer program which obtains waveguide mode solutions for very low frequencies and low frequencies (VLF/LF) is described. The program allows for multiple homogeneous segments to be specified, allowing for consideration of variations in the earth-ionosphere waveguide. Path geometry and geophysical parameters can be computed by the program. Ionospheric disturbances due to man-made or naturally occurring events can also be modeled using the program.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
J.A. Ferguson	(619) 225-2974	Code 544

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

DD FORM 1473, 84 JAN

CONTENTS

APPENDIX

FIGURES

TABLES

INTRODUCTION

The model of longwave propagation developed at the Naval Ocean Systems Center (NOSC) is based on a waveguide mode formulation. To determine signal levels in this approach the basic problem is to obtain the modal solutions to the specific waveguide under consideration (Pappert et al., 1970, Morfitt and Shellman, 1976), where complicated propagation paths are divided into horizontally homogeneous segments. The parameters of the segments are determined by the earth's ground conductivity, the magnitude and orientation of the geomagnetic field with respect to the direction of propagation, and the state of the ionosphere. Well-known computer programs which make the necessary calculations for a single set of propagation parameters are "WAVEGUID" (Pappert et al., 1970) and "MODESRCH" (Morfitt and Shellman, 1976). The WAVEGUID computer program and related programs are described in a series of reports and familiarity with the important elements of this series is assumed (Pappert, Gossard and Rothmuller, 1967; Sheddy et al., 1968; Pappert, Moler and Shockey, 1970; Morfitt and Shellman, 1976). This report describes a modified version of that program designed to supply calculations of long wavelength propagation along segmented propagation paths. This program is called the Segmented Waveguide (SW).

The computer program obtains waveguide mode solutions for very low frequencies and low frequencies (VLF/LF). The program allows for multiple homogeneous segments to be specified, allowing for consideration of variations in the earth-ionosphere waveguide. Path geometry and geophysical parameters can be computed by the program. Ionospheric disturbances due to man-made or naturally occurring events can also be modeled using the program.

Essential features of this program include:

- Automatic segmentation of the propagation path
- Allowance for presegmentation of the propagation path
- Allowance for variation of the ionosphere along that path

The program operates on propagation paths defined by a transmitter location and either a direction or a receiver location. A propagation path is defined as the great circle on a spherical earth. Variation of the geophysical parameters are to be expected along realistic paths. The diurnal condition in large part determines the significance of the other parameters. For instance, under daytime conditions the effect of variation of the geomagnetic field along a path is usually not significant to the resulting mode parameters. The program incorporates routines for calculating the parameters of the geomagnetic field and for selecting the ground conductivity at any point on the earth's surface.

Each of the path segments is treated as a horizontally homogeneous planar waveguide. Earth curvature is introduced by use of a modified refractive index. A set of possible solutions to the waveguide mode equation must be input. Each of these solutions is processed by an iteration routine. Each iteration requires computation of ionospheric and ground reflection coefficients. Calculation of the ionospheric reflection coefficients requires integration of the coefficients through the ionosphere. An approximate formulation may be used which requires a secondary set of complex angles be specified by the user. In that case, the ionospheric reflection coefficients are calculated for the secondary set of angles. These coefficients are used to interpolate the ionospheric reflection coefficients during the iteration of the primary set of possible solutions. This interpolation procedure requires much less computation time than does the more exact procedure.

The set of solutions for the first homogeneous segment must be input by the user. The program uses the results of up to three consecutive segments to extrapolate the solutions for successive segments. This reduces the number of iterations which are required for subsequent segments and allows for the tracing of mode solutions through a wide range of path variations.

The primary output of "SW" is data which may be used in mode summing programs. The strength of the electromagnetic field along the path can be obtained with either of two mode conversion models, one denoted "FULLMC" (Pappert and Shockey, 1972) and the other denoted "FASTMC" (Ferguson and Snyder, 1980). "FULLMC" does integrations through the ionosphere prior to calculating mode conversion coefficients and can be quite slow in execution time whereas "FASTMC" avoids the transionospheric integrations by use of approximations and runs quite quickly.

PROGRAM CONTROL

Program execution is controlled by strings containing mnemonic words. These strings and the sequence of operations initiated by them are described below. These control strings, as well as variable names and names of subroutines, appear in upper case. For clarity, the control strings are enclosed in single quotes and names of subroutines are enclosed in double quotes. Table 1 summarizes these control strings.

Table 1. Summary of control strings.

ID	run identification
NAME	general NAMELIST data
EIGEN	alternate EIGEN list input
PROFILE i	ionospheric specie profiles (i is 1 or 3)
COLFREQ	ionospheric collision frequency profiles
COORD	automatic segmentation of propagation path
PRESEG	presegmentated propagation path
QUIT	end of job

'ID' indicates that the next line of data to be read is a general identification for the path under consideration. This identification appears in the printout. It is also written to the mode parameter output file.

'NAME' initiates the reading of general program data via the NAMELIST: DATUM. Table 2 lists all variables, their type (I indicates integers, R indicates real (floating point) variables, C indicates complex variables), their units where applicable, and their initial values. If a variable defines an array, then the dimension of the array follows the name in parentheses. The NAMELIST input format is quite flexible but requires that column 1 be blank. Variable names are followed by an equal sign and then by the value of the variable. Array variable names may be followed by a string of values separated by commas and/or spaces. Embedded blanks are not allowed in the variable names. Variable types must be considered; for example, values for integer variables may not contain decimal points, but values for real variables do not have to have decimal points. Logical variables may be specified with any of '.true.' '.t.' 't' '.false.' '.f.' 'f'. The values of character variables must be enclosed within quotes. The first record of the NAMELIST input must contain ' &name ' where 'name' is the

NAMELIST name (in this case, DATUM). The end of the NAMELIST is indicated by '&end'. Some of the values to be found in the following text are of the form A(N) to indicate A to the power N.

'EIGEN' allows for input of the list of trial solutions to be made from a file. The format of the input is the same as with NAMELIST. The control string is followed by the name of the file (starting in column 9) containing the NAMELIST data. The source of these input solutions could be a previous run with the program or from one of the automatic mode searching programs such as the one described by Morfitt and Shellman (1976).

'PROFILE i' initiates reading of the ionospheric charged particle profile data used to model the upper boundary of the earth-ionosphere waveguide. The value of i indicates the number of ionosperic species to be used and it must have one of two possible values: 1 is for electrons only and 3 is for electrons and ions. If i is not specified, a value of 1 is assumed. The use of i is shown below. The 'PROFILE i' string is immediately followed by a single line of identification for the profile. The profile is input starting at the top of the ionosphere using a formatted input. Each line contains the height in km, the electron density at that height in electrons per cubic centimeter, and if ions are to be considered, the positive ion density at that height in ions per cubic centimeter. The height is in columns 1-7 and the electron and ion densities are required to be in columns 14-21 and 24-31, respectively. The end of the profile is indicated by a dummy height with value less than zero. A maximum of 50 heights may be used. If i is 1, then only the electron density is read. Consequently, only the electron density need be present in the data. If i is 3, then the electron and positive ion densities are read and the negative ion density is computed by subtracting the electron density from the positive ion density (to preserve charge neutrality).

In the integration of the reflection elements through the ionosphere the program interpolates exponentially between input heights. The profile should contain sufficient data to define the ionospheric structure with height. For example, an exponential profile should consist of only the top and bottom heights and densities. Many regularly spaced heights tend to slow the integration.

A purely exponential conductivity profile (electrons only) may be input via the NAMELIST variables BETA and HPRIME.

Additional specie parameters are needed for the waveguide mode computations and are described below.

'COLFREQ' initiates reading of an ionospheric collision frequency profile. This allows use of nonexponential collision frequencies. The 'COLFREQ' string is immediately followed by the collision frequency profile, starting with the highest height and ending with a dummy height of value less than zero as in the case of specie profile described above. These heights need not match those used under 'PROFILE i'. The format of the data is the same as used for 'PROFILE i' except that collision frequencies for all species must be input since the negative ion collision frequency cannot be simply computed from the other two. If only electrons are being used, then only that collision frequency need be present. As with the specie profiles, the program interpolates exponentially between input heights.

An exponential collision frequency specification may be input via the NAMELIST variables EXPNU and COEFNU.

'COORD' initiates automatic segmentation of the propagation path. This string must be placed after all pertinent data have been read. This option is best applied to simple cases such as all daytime. The basic input consists of the path specification, the environment, and the starting mode solutions.

'PRESEG' allows for previously determined segments to be used along the propagation path. This option requires most of the same inputs as 'COORD' except that the user supplies the distances at which segments begin. At each segment the user has the option of specifying the parameters of the geomagnetic field, the ground and the ionosphere.

CALCULATION OF MODE PARAMETERS

The inputs to the mode equation computations are supplied by geophysical routines and/or by the user. The subroutine which controls the calculations is "WVGUID". Most user supplied data values are input to the program via NAMELIST. These parameters are summarized in table 2 and are discussed in more detail below. In table 2 the data types are Integer (I), Real (R), and Complex (C).

Table 2. NAMELIST Inputs.

Varible	Type	Default	Description
FREQ	R	0.0	Frequency in kHz
RHO	R	0.0	Distance from the transmitter of the current segment in Mm.
AZIM	R	0.0	Magnetic azimuth angle in degrees east of magnetic north.
CODIP	R	0.0	Magnetic co-dip angle in degrees.
MAGFLD	R	0.0	Intensity of the geomagnetic field in Webers/square meter.
SIGMA	R	4.64	Ground conductivity in Siemens/meter.
EPSR	R	81.0	Dielectric constant of the ground.
BETA	R	0.0	Slope of the exponential profile in km(-1)
HPRIME	R	0.0	Reference height of the exponential profile in km.
TLONG,TLAT	R	0.0,0.0	Transmitter coordinates in degrees
RBEAR	R	720.0	Geographic bearing of the path in degrees
RLONG,RLAT	R	0.0,0.0	Receiver coordinates in degrees
DRMIN	R	0.125	Minimum distance step size in Mm
DRMAX	R	0.5	Maximum distance step size in Mm
DMAX	R	20.0	Maximum distance in Mm
YEAR	I	0	Year
MONTH	I	0	Month number with January being 1
DAY	I	0	Day of the month
GMT	R	0.0	Greenwich meridian time in hours
NPRINT	I	1	Flag used to control the amount of print out
NPROF	I	1	Flag to indicate which form of the profile is to be used

MIDPNT	I	0	Flag to indicate that only the midpoint is to be considered
IGCD	I	0	Flag to indicate that the computed distance between the transmitter and receiver is to be used
IGND	I	0	Flag to indicate that the ground conductivity is to be determined from the ground map
MDIR	I	0	Flag used to reverse the direction of the magnetic field
EIGEN(60)	C	0.0,0.0	Initial solutions for the waveguide modes in degrees.
TLIST(30)	C	0.0,0.0	Angles where the reflection coefficients are computed for the inexact interpolation routine in degrees
DTHETA	C	0.01,0.001	Change in the mode solution used to define the iteration in degrees
LUB	C	0.05,0.005	Tolerance used to test the differential change in the mode solution. Used to stop the iteration
DEIGEN	C	0.05,0.005	Tolerance used to define duplicate modes in degrees
FTOL	R	1.0	Tolerance used to determine if the mode equation has been satisfied by the solutions
THTINC	R	0.5	Maximum change in degrees of either the real or imaginary part of the mode solution from one iteration to the next
MAXITR	I	7	Maximum number of iterations to attempt
ALPHA	R	3.14x10(-4)	The earth curvature correction factor in km(-1)
H	R	50.0	Height at which the modified refractive index is unity in km
D	R	0.0	Height at which the integration through the ionosphere is stopped in km
PREC	R	2.0	Factor which controls the precision of the reflection coefficient integration
WR0	R	2.5x10(5)	Value of omega sub r used to define the reference height
ATNMAX	R	50.0	Maximum attenuation rate of modes to be retained in dB/Mm
DEBUG	I	0	Flag used to generate additional printout for debugging purposes
TYPITR	I	0	Flag used to define the form of the mode equation
RPOLY	I	1	Flag used to define the reflection coefficient calculation
NRTLST	I	5	Number of points to use in the interpolation of the reflection coefficients during inexact iterations
LUNIT7	I	7	Logical unit number to which the mode parameters data are output
CHARGE(3)	R	-1.0, 1.0,-1.0	Charge of the ionospheric species
MRATIO(3)	R	1.0, 2*5.8x10(4)	Ratio of the mass of the ionospheric species to that of electrons
COEFNU(3)	R	1.816x10(11) 2*4.54x10(10)	Collision frequency of the ionospheric species at the ground in collisions/sec
EXPNU(3)	R	-0.15, 2*-0.15	Exponential slope of the collision frequency in km(-1)

The radio frequency in kHz is specified by the variable FREQ. The input to subroutine "WVGUID" includes an ionospheric profile. The variable NPROF controls which ionospheric profile is to be used. A value of 0 indicates that the profile input via 'PROFILE i' is to be used. A value of 1 indicates that an exponential electrons only profile is to be used. The profile is specified by the exponential slope BETA in km(-1) and a reference height HPRIME in km (Wait and Spies, 1964). A value of 2 for NPROF is used to indicate that a series of profiles will be input. This option only applies to 'PRESEG'. The profiles are to be input using the same format as described under 'PROFILE i', including the control string. There must be a profile for each segment.

Additional specie parameters are needed. The number and order of these specie parameters must be consistent with the charged particle densities of 'PROFILE i'. The charges of the species are input as CHARGE (i.e., CHARGE = -1, +1, -1). The masses of the species relative to that of an electron are input as MRATIO. The collision frequencies may be defined with 'COLFREQ' (nonexponential) or with the variables COEFNU in collisions per second and EXPNU in km(-1). The collision frequency v at an altitude z is then defined by

$$v = \text{COEFNU} * \exp(\text{EXPNU} * z)$$

where z is in km.

Parameters of the geomagnetic field are specified by AZIM, the angle between magnetic north and the direction of propagation in the horizontal plane measured in degrees east of north, CODIP, the magnetic co-dip angle measured from the vertical (i.e., the north pole has a CODIP of 0), and MAGFLD, the magnetic intensity in Webers per square meter or in Gauss. The magnitude of MAGFLD is tested. If it is greater than 10(-2) then the input value is assumed to be in Gauss and is multiplied by 10(-4).

MDIR is a flag that when set to 1, causes the direction of propagation as input to be reversed. This allows for development of a data set appropriate to examining transmitter deployment.

Ground conditions are specified by SIGMA, the conductivity in Siemens/m, and EPSR, the relative dielectric constant.

The presegmentation option allows the magnetic field and ground parameters to be varied by the user.

The correction for earth curvature is controlled by ALPHA in km(-1) which is defined as 2 over the radius of the earth. For a curved earth ALPHA is 3.14x10(-4) km(-1) and for a flat earth ALPHA is 0.

Ionospheric altitude parameters are H, which is the height in km at which the modified refractive index is unity and is the height to which the mode solutions are referenced; D, which is the height in km below which ionospheric effects can be ignored. D must be equal to or greater than the bottom height of the ionospheric profiles used. It is usually sufficient to choose H equal to D. The choice of D and H is also discussed by Pappert et al. (1967).

The trial eigenangles follow the variable name EIGEN which is a complex variable. Up to 60 eigen angles may be input. If little is known about the expected solutions for a given set of conditions, a set of approximate solutions may be obtained using a TLIST. The TLIST is a list of as many as 30 complex angles which are used to set up an interpolation matrix of the ionospheric reflection elements (Sheddy et al., 1968). The program then uses this matrix to interpolate reflection elements during the iterative process used to obtain mode solutions. These solutions are referred to as "inexacts" in order to distinguish them from the more accurate solutions using integrated reflection coefficients. The variable NRTLST determines the maximum number of TLIST angles used in each interpolation. During the inexact iteration process, the program computes the magnitude of the complex difference between the current value of the solution and each of the TLIST angles. The program orders the TLIST angles from the smallest to the largest difference and selects the first NRTLST of them to be used in the interpolation. This improves the accuracy of the interpolated reflection coefficients and reduces the number of terms used.

If more than 30 EIGENs are input, then the program sorts the angles according to their attenuation rate and deletes those with attenuation rates greater than a user specified maximum. The initial value of this maximum is ATNMAX. If there are still more than 30 angles, then the maximum attenuation rate is reduced by 5 dB/Mm and the input list is sorted again. This process is repeated until there are less than 30 angles in the list.

If the number of EIGEN or TLIST inputs varies from one NAMELIST to the next, then each EIGEN or TLIST list should be terminated with a zero. If RPOLY is not 0 and the first value of TLIST is 0, then TLIST is set equal to the first 30 EIGENs.

The Newton-Raphson iteration process, used to find the eigenangles which satisfy the modal equation is described by Sheddy et al. (1968). Iteration is performed for each input EIGEN. The iteration stops when the maximum number of iterations (MAXITR) is exceeded or when the change in the real and imaginary parts of the solution is calculated to be less than the real and imaginary parts of LUB, respectively.

The type of solution obtained is determined, in part, by RPOLY, which can have three values: 0 for exact solutions only, 2 for inexact (approximate) solutions only, and 1 for inexacts computed and used as inputs to obtain exact solutions. The use of RPOLY equal to 1 is described more fully below.

The flag TYPITR is used to obtain vertically polarized modes only (TYPITR equal to 1) or horizontally polarized modes only (TYPITR equal to 2). It is physically meaningful to apply this option only for nearly isotropic conditions, no magnetic field (MAGFLD set to 0), or east to west and west to east propagation at the geomagnetic equator (CODIP is 90 and AZIM is 90 or CODIP is 90 and AZIM is 270).

To ensure consistent mode sums and eliminate redundant solutions, each exact and inexact EIGEN solution is tested for several conditions. The first is that the imaginary part of the solution must be less than zero in order to have attenuating modes. The second is that the magnitude of the modal equation must be less than FTOL. This parameter is tested only for final solutions and only if the number of iterations required to obtain the solution is greater than or equal to MAXITR. It is generally true that if the iteration stops because the change in the mode solution is less than LUB, then the value of the mode equation is small. There are instances in which the test on FTOL will still fail. Consequently, the value of FTOL is set very high in order to allow the

program to continue execution. In some cases the user may want to modify the default value in order to perform special tests. If RPOLY is 1, the inexacts are treated as intermediate results. The third test is that the value of the EIGEN solution must be different from all previous solutions by an amount DEIGEN which is input as a complex number. The real and imaginary parts of DEIGEN are the tolerances for the real and imaginary parts of the EIGEN solutions, respectively. If one of the above tests results in a mode being dropped from the list of solutions, then the program follows the procedures outlined below under the discussion of mode tracing.

Subroutine "WVGUID" computes and prints attenuation rate in dB/Mm, phase velocity relative to the speed of light, the magnitude, and phase of Wait's excitation factor (Wait, 1962) at the ground in dB and radians.

The headings for the number of iterations to go from the input angle to the final solution, the final solution, the magnitude of the modal equation, and the magnitude of the polarization mixing ratio are printed as ITER, EIGEN, MAG F, and MAG P, respectively. The attenuation rate, phase velocity relative to the speed of light, magnitude, and phase of Wait's excitation factor, and the final solution references to the ground are printed under the headings of ATTEN, V/C, WAIT'S EXC, and THETA', respectively.

The parameters YEAR, MONTH, DAY, and GMT are used only to pass the values to the output files. These parameters are useful for helping to identify the output data to programs which may use this information.

PATH CALCULATIONS

These calculations are controlled by "GCPATH". They can be divided into three classes. The first automatically computes geometry and geophysical parameters which are obtainable from just the location of a point on the propagation path or uses presegmented distances and geophysical parameters. In addition, the program extrapolates the EIGEN list and TLIST so as to trace modes along a path.

PATH GEOMETRY

All geometry calculations are for a spherical earth. Inputs to this portion of the program consist of transmitter and receiver locations, path length, and path increments. Transmitter and receiver longitude and latitude are input with TLONG and RLONG and TLAT and RLAT. The convention used in the program is east longitude and south latitude are negative. An alternate input for the receiver position is its bearing, RBEAR, in degrees east of north. In the execution of the program RBEAR is tested. If RBEAR is 720, then the program uses RLONG and RLAT to define the path. If RBEAR is not 720, then it and the input path length, DMAX, are used to define the path. DMAX is specified in megameters and must be less than or equal to 20. If RLONG and RLAT are used, there are two path lengths possible. If the parameter IGCD is equal to 1, the path length is set equal to the computed short great circle distance between the transmitter and receiver. If IGCD is equal to 0, the path length is unchanged from what was input (DMAX). If the path bearing is input, the path length is always DMAX.

The starting value of the distance from the transmitter is input as RH0 and is in Mm. The path increments are controlled by the mode tracing results and the variables DRMIN and DRMAX which are all in units of Mm. The procedures of the values are described as follows.

The geomagnetic field is computed at the first path point defined by RHO and LAT at the beginning of each path segment. The ground conductivity and relative dielectric constant are specified by the user through SIGMA and EPSR via NAMELIST or by searching the ground map. If IGND is 0, the ground map is not searched and ground conditions are assumed constant, as input, for the entire path for 'COORD' or as varied by the user for 'PRESEG'. If IGND is 1, the DECO-NRL 10 level ground conductivity map (Hauser, Garner, and Rhoads, 1969) is searched for the appropriate values of SIGMA and EPSR at the beginning of each path segment.

If the value is assumed that the entire propagation path can be described adequately by the conditions at the midpoint of the path, then the path conditions at that point can be obtained if MIDPNT is set to 1. The subsequent modal calculations will then be for the midpoint conditions.

At the transmitter and at the end of each path segment, the parameters to be used in the "WVGUID" calculations are printed next to the heading, PROPAGATION PATH PARAMETERS, as described below. In addition, the distance in megameters from the transmitter, the coordinates and the geographic bearing of the path at the current point are printed under the headings RHO, LAT, and BEAR, respectively. If the midpoint option is being used, then the above information for the midpoint is printed.

PRESEGMENTATION

In some instances, user segmentation of the propagation path is desired. The control string 'PRESEG' allows arbitrary segmentation of the path. This is accomplished by a succession of data lines in list directed format containing values for path distance in Mm, AZIM, CODIP, MAGFLD, SIGMA, EPSR, BETA, and HPRIME, respectively. List directed input is accomplished by entering values separated by commas or spaces. There must a data entry for each request in the input list. If a value is not to change from one data line to the next, then the value need not be entered but its omission must be indicated by a pair of commas. The first value of path distance need not be zero. The presegmentation is terminated by a distance value of 40.

If NPROF is 0, the ionospheric profile is constant for the path and is defined by 'PROFILE i'. If NPROF is 1, then the values of BETA and HPRIME on the presegmentation data lines are examined. If BETA is zero, then the previously defined values of BETA and HPRIME are used. The latter may be input via NAMELIST so that a constant ionosphere for the path can be obtained by using NAMELIST input. If either of the values of BETA and HPRIME are to change, both must be input. If NPROF is 2, then each presegmentation data line must have a corresponding 'PROFILE i' profile specification on logical unit 3.

The following conventions are used for using the values of the presegmentation data. If the value of MAGFLD is zero or blank, then the magnetic field parameters are calculated. If a nonzero value is specified, then all of the magnetic field parameters are taken from the presegmentation data. If constant magnetic parameters are desired along the path, the values must be specified on each presegmentation data line. If the value of SIGMA is not entered, then the previously specified value of SIGMA and EPSR are used. Otherwise, the values of these parameters are taken from the presegmentation data line. Constant values for the whole path may be specified via NAMELIST or in the first presegmentation card. If IGND is 1, then the ground

map is searched and the values of SIGMA and EPSR on the cards are ignored. if BETA is not entered, the currently defined values are used for the electron density profile. Even if only one value in the pairs SIGMA, EPSR, and BETA, HPRIME is to be changed, both values must be specified.

MODE TRACING

Efficient computation of mode parameters along the propagation path is best achieved by using RPOLY set to 1, which will be assumed for the rest of this discussion. At the first point on the path, solutions are best obtained by using a TLIST composed of angles which are believed to be approximately correct and an EIGEN list of many regularly spaced angles such as 88, -1, 87, -1, 86, -1, etc. Alternatively, the EIGEN list should be the list of approximately correct solutions with the TLIST set to zero or the 'EIGEN' control string could be used to specify solutions from some other source. The program computes inexact solutions for the conditions at the first point on the path. After exhausting the EIGEN list, obtaining inexact solutions, and deleting of those failing the tests discussed above, exact solutions are computed using the results of the inexacts.

Now the discussion must be separated for the two-path segmentation options. For the 'COORD' option, the second point on the path is DRMIN from the transmitter. For this point, the final solutions for the first point are placed in both TLIST and EIGEN and the same process of calculation of inexacts and exacts is repeated. If DRMIN is not too far from the transmitter and/or the geophysical parameters do not change too much, then this step in the extrapolation process is quite efficient. Now the program has two sets of final solutions and makes a linear extrapolation for TLIST and EIGEN angles for the third point on the path which is twice DRMIN from the transmitter. The sequence of calculations for the inexact and exact solutions is repeated. For the fourth and all subsequent points on the propagation path, the program uses the previous sets of final solutions to make second order extrapolations for TLIST and EIGEN angles. The distance increments are chosen as described below.

As the program steps out along the propagation path, modal solutions may be lost or removed. First, a mode may be lost in the screening process in subroutine "WVGUID" as described above. At the first point on the path, modes may be overlooked simply because of lack of adequate trial solutions and/or more than one EIGEN input resulting in the same final solution, perhaps due to closely spaced input EIGEN values. If computations are being made at the transmitter or at the midpoint, it is acceptable to lose a solution from the input EIGEN list. At all other points, when a mode is lost execution terminates in "WVGUID". After the tests on the solutions in "WVGUID" are completed at the first point on the path, the program assumes that it has a complete set of modes. After this set is established, solutions may be acceptably removed only in the extrapolation subroutine, "EXTRAP". The solutions produced by "WVGUID" for the current segment are used to compute attenuation rates. Those solutions whose attenuation rate exceeds ATNMAX are deleted from the list. The location of the solution in the set is marked and its removal is indicated by a blank line in the printout of solutions produced by subsequent "WVGUID" calculations.

If a mode is lost during "WVGUID" calculations, the path point is moved back to about halfway between the current point (where a solution was lost) and the previous point (where all solutions were obtained). The actual distance depends on the current value of the distance increment. If the increment is greater than DRMIN, then the

new increment is chosen that it is an integral multiple of DRMIN and is less than or equal to half the previous increment. Geophysical parameters at the new path point are computed, the EIGEN list is revised by "EXTRAP", and "WVGUID" calculations are repeated.

If no modes are lost and the number of iterations required to obtain the solutions is less than or equal to half of MAXITR, then the distance increment is increased by DRMIN. This increase in the distance increment can continue until the path increment is equal to DRMAX. If no modes are lost and the number of iterations required to obtain the solutions is greater than half of MAXITR, the path increment is decreased by DRMIN.

If modes are lost and the separation between the previous point (for which all modes were found) and the current point (for which modes were lost) is less than or equal to DRMIN, then the distance increment is halved. The geophysical parameters for the new path point are linearly interpolated using the parameters of the two points at which geophysical parameters were computed, the EIGEN list is revised by "EXTRAP", and "WVGUID" calculations are repeated. Solutions obtained for interpolated path points are not saved. They are used only to trace the mode solutions between the points for which the geophysical parameters are computed.

If no modes are lost and the number of iterations required to obtain the solutions is less than or equal to half of MAXITR, then the distance increment is doubled. This increase in the distance increment can continue until the path increment is equal to the distance to the end of the interpolation interval. If no modes are lost and the number of iterations required to obtain the solutions is greater than half of MAXITR, the path increment is halved. If the new distance increment is less than 15 km, then the program aborts.

For the 'PRESEG' option, the distance increment is controlled by the intervals between the presegmented distances. When modes are lost, the interpolation procedure for cases in which the backup interval is less than DRMIN described above is followed.

OUTPUT

Mode parameters from the program are written to the logical unit whose numerical value is LUNIT7. The first line of data written contains the transmitter location, path bearing, and the date and time, as input through NAMELIST. The identification which followed 'ID' is written next. If no identification was specified with 'ID', this line of data is blank. The identification is followed by a sequence of lines at each output distance.

The first line of data at each such distance contains the distance, frequency, AZIM, CODIP, MAGFLD, SIGMA, EPSR, and the reference height of the ionospheric profile. In descriptions of other programs, this first line of data at each distance is referred to as the RFACMSET header. This header line is followed by pairs of data lines, one pair for each mode. The quantities in these data lines are the mode solution as a complex angle in degrees, a flag, T1, T2, T3, and T4. The flag and the parameters T1, T2, T3, and T4 are described in detail by Ferguson and Snyder (1980). The last line of data at each output distance is blank. These data are suitable for use in "FASTMC" (Ferguson and Snyder, 1980).

If the program fails because of some problem at the first path point, it writes 'Failure at RHO 1' to logical unit 90. Otherwise, it writes the distance of the last point for which "WVGUID" successfully completed. If the end of the path is reached, then this distance is output as 40.

SAMPLE INPUT

Sample input files are shown in figures 1 and 2. In the first sample (figure 1), the path is to be run for all nighttime conditions assuming all seawater ground. The EIGEN list for the transmitter is input directly and the automatic path segmentation is to be used.

```
id
Sample run
name
 &datum freq=23.4 h=50 d=75 lunit7=7 atnmax=50
 tlong=150 tlat=20 rbear=10 dmax=10
 lub=.005 .0005 dtheta=.01 .001
 deigen=.05 .005 thtinc=.05
 beta=.43 hprime=87
 eigen= 85.678 -0.206 84.595 -0.688 81.806 -0.609 81.027 -0.255
        77.653 -0.791 77.023 -0.269 73.199 -0.825 72.980 -0.300
        68.926 -0.264 68.599 -0.955 64.751 -0.213 63.907 -1.144
        60.457 -0.183
 &end
coord
```

Figure 1. Sample input using COORD option.

The second example (figure 2) is a much more complicated case. It is for the same path of the first sample, but the ionosphere is to be varied according to the diurnal conditions along the path for July 15 at 1612Z. The transition from night to day has been modeled as five steps starting with BETA at 0.30 and HPRIME at 74 and ending with BETA at 0.43 and HPRIME at 87. In addition, the ground conductivity for the last profile takes on three values: 4, 10(-2), and 10(-3). In order to improve the efficiency of the mode tracing, additional segmentation has been performed so that each ground conductivity is processed separately. The segmentation does not produce final output that is monotonically increasing in distance from the transmitter. The necessary ordering of the segments must be performed by editing the final output file or by a user supplied program. The initial mode solutions for each segment have been already calculated and stored in a set of files named XMTR202.MFx where x ranges from 0 to 6.

```
id
Sample run
name
 &datum freq=23.400 h=50. lunit7=7 atnmax=50.
 lub=0.005 0.0005 dtheta=0.010 0.0010
 deigen=0.050 0.0050 thtinc=0.05
 year=84 month= 7 day=15 gmt=16.2
 tlong= 158.150 tlat= 21.417 dmax=4
 rbear=202.0 &end
eigen   xmtr202.mf0
preseg
 0.000,190.6,  50.8,0.350,4.E+00,81.,0.30,74.0,
40,0,0,0,0,0,0,0,0,
eigen   xmtr202.mf1
preseg
 0.500,190.8,  56.8,0.337,4.E+00,81.,0.32,76.2,
40,0,0,0,0,0,0,0,0,
eigen   xmtr202.mf2
preseg
 0.960,190.9,  63.1,0.329,4.E+00,81.,0.34,78.3,
40,0,0,0,0,0,0,0,0,
eigen   xmtr202.mf3
preseg
 1.040,190.9,  64.2,0.327,4.E+00,81.,0.37,80.5,
 1.240,190.9,  67.3,0.325,4.E+00,81.,0.37,80.5,
40,0,0,0,0,0,0,0,0,
eigen   xmtr202.mf4
preseg
 1.340,190.9,  68.8,0.324,4.E+00,81.,0.39,82.7,
 1.540,190.9,  72.1,0.323,4.E+00,81.,0.39,82.7,
40,0,0,0,0,0,0,0,0,
eigen   xmtr202.mf5
preseg
 1.640,190.9,  73.8,0.322,4.E+00,81.,0.41,84.8,
 1.820,190.9,  76.8,0.322,4.E+00,81.,0.41,84.8,
40,0,0,0,0,0,0,0,0,
eigen   xmtr202.mf6
preseg
 1.940,190.8,  78.9,0.322,4.E+00,81.,0.43,87.0,
 2.120,190.8,  82.2,0.323,4.E+00,81.,0.43,87.0,
 2.300,190.7,  85.5,0.324,4.E+00,81.,0.43,87.0,
 2.480,190.6,  88.8,0.326,4.E+00,81.,0.43,87.0,
 2.660,190.6,  92.1,0.329,4.E+00,81.,0.43,87.0,
 2.840,190.5,  95.5,0.333,4.E+00,81.,0.43,87.0,
 3.020,190.4,  98.9,0.337,4.E+00,81.,0.43,87.0,
40,0,0,0,0,0,0,0,0,
```

Figure 2. Sample input using PRESEG option.

REFERENCES

Ferguson, J. A., and F. P. Snyder, "Approximate VLF/LF Waveguide Mode Conversion Model", NAVOCEANSYSCEN Technical Document 400, November 1980.

Hauser, J. P., W. E. Garner, and F. J. Rhoads, "A VLF Effective Ground Conductivity Map of Canada and Greenland With Revisions Derived From Propagation Data", NRL Report 6893, March 1969.

Morfitt, D. G., and C. H. Shellman, "MODESRCH, An Improved Computer Program for Obtaining ELF/VLF/LF Mode Constants in an Earth-Ionosphere Waveguide", DNA Interim Report No. 77T, October 1976.

Pappert, R. A., E. E. Gossard and I. J. Rothmuller, "A Numerical Investigation of Classical Approximations Used in VLF Propagation," *Radio Science*, Vol 2 (New Series), No. 4, April 1967.

Pappert, R. A., W. F. Moler and L. R. Shockey, "A FORTRAN Program for Waveguide Propagation Which Alows for Both Vertical and Horizontal Dipole Excitation", DASA Interim Report No. 702, June 1970.

Pappert, R. A. and L. R. Shockey, "Mode Conversion Program for an Inhomogeneous Anisotropic Ionosphere", NELC Interim Report No. 722, May 1972.

Pappert, R. A. and L. R. Shockey, "A Simplified Mode Conversion Program for VLF Propagation in the Earth-Ionosphere Waveguide", DASA Interim Report No. 751, October 1974.

Sheddy, C. H., R. A. Pappert, Y. A. Gough, and W. F. Moler, "A FORTRAN Program for Mode Constants in an Earth-Ionosphere Waveguide", DASA Report 683, May 1968.

Wait, J. R., *Electromagnetic Waves in Stratified Media*, Pergamon Press, New York, p. 221, 1962.

APPENDIX: LISTING OF THE PROGRAM

```
0001    c     SW:  SEGMENTED WAVEGUID
0002    c
0003          include 'common1.for/list'
0004  1 c
0005  1       common/input/freq,rho,azim,codip,magfld,sigma,epsr,beta,hprime,
0006  1      $       hprout
0007  1       common/path/pathid,tlong,tlat,rlong,rlat,rbear,dmax,drmin,drmax,
0008  1      $       year,month,day,gmt,nprint,nprof,npath,igcd,ignd,mdir,lost,
0009  1      $       lunit7,lx
0010  1       common/ionosp/htlist(50),lnlist(50,3),hclist(50),cflist(50,3),
0011  1      $       charge(3),mratio(3),nrspec,lhtmx,lhtmn,lht,mhtmx,mhtmn,mht
0012  1 c
0013  1       character*80 pathid
0014  1       integer year,day
0015  1       real*4 freq,rho,azim,codip,magfld,sigma,epsr,beta,hprime,hprout,
0016  1      $       tlong,tlat,rlong,rlat,rbear,dmax,drmin,drmax,gmt,
0017  1      $       htlist,lnlist,hclist,cflist,charge,mratio
0018  1 c
0019          include 'common1.ini/list'
0020  1 c
0021  1 c     initialize common1
0022  1       data freq/0./,rho/0./,azim/0./,codip/0./,magfld/0./,
0023  1      $     sigma/4.64/,epsr/81./,beta/0./,hprime/0./,
0024  1      $     tlong/0./,tlat/0./,rlong/0./,rlat/0./,rbear/720./,
0025  1      $     dmax/20./,drmin/.125/,drmax/.5/,mdir/0/,
0026  1      $     year/0/,month/0/,day/0/,gmt/0./,nprint/1/,nprof/1/,
0027  1      $     igcd/0/,ignd/0/,mdir/0/,lunit7/7/,
0028  1      $     charge/-1.,1.,-1./,mratio/1.,2*58000./,nrspec/1/
0029  1 c
0030          include 'common2.for/list'
0031  1 c
0032  1       common/wg  in/elist(2,30),tlist(2,30),dtheta(2),lub(2),deigen(2),
0033  1      $       thtinc,ftol,maxitr,alpha,h,d,prec,wr0,atnmax,debug,typitr,
0034  1      $       rpoly,nrtlst
0035  1       common/wg out/tp(30),tterm(4,30),nterm(30),mode(30),modes,nmds
0036  1 c
0037  1       complex*8 tp,tterm,dthta
0038  1       integer debug,typitr,rpoly
0039  1       real*4 elist,tlist,dtheta,lub,deigen,thtinc,ftol,alpha,h,d,prec,
0040  1      $       wr0,atnmax
0041  1 c
0042  1       equivalence (dtheta,dthta)
0043  1 c
0044          include 'common2.ini/list'
0045  1 c
0046  1 c     initialize common2
0047  1       data elist/60*0./,tlist/60*0./,
0048  1      $     dtheta/.01,.001/,lub/.05,.005/,deigen/.05,.005/,thtinc/.5/,
0049  1      $     ftol/1000./,maxitr/7/,alpha/3.14e-4/,h/50./,d/0./,prec/2./,
0050  1      $     wr0/2.5e5/,atnmax/50./,debug,typitr/0,0/,rpoly/1/,nrtlst/5/
0051  1 c
0052    c
0053          namelist/datum/freq,rho,azim,codip,magfld,sigma,epsr,beta,hprime,
0054         $         tlong,tlat,rlong,rlat,rbear,dmax,drmin,drmax,
0055         $         year,month,day,gmt,nprint,nprof,midpnt,igcd,ignd,mdir,
0056         $         lunit7,charge,mratio,coefnu,expnu,
0057         $         eigen,tlist,dtheta,lub,deigen,thtinc,ftol,maxitr,
```

SW$MAIN

```
0058          $         alpha,h,d,prec,wr0,atnmax,typitr,rpoly,prtlst
0059     c
0060           complex theta
0061           character* 8 branch
0062           character*40 fname
0063           character*80 bcd
0064     c
0065           dimension coefnu(3),expnu(3),eigen(2,60)
0066     c
0067     c     Initialize MAIN
0068           data nuflag/0/,coefnu/1.816e11,2*4.540e09/,expnu/3*-.15/,
0069          $     eigen/120*0./,midpnt/0/
0070     c
0071     c     Unit:       Useage:
0072     c         2               input  of alternate eigen list
0073     c         3               input  of profiles along the path
0074     c         lunit7          output of mode parameters along the path
0075     c
0076     c
0077     10    read(5,1000,end=999) bcd
0078           print 1001,bcd
0079           branch=bcd(1:8)
0080           if(branch .eq. 'id      ' .or. branch .eq. 'ID      ') go to 20
0081           if(branch .eq. 'name    ' .or. branch .eq. 'NAME    ') go to 100
0082           if(branch .eq. 'eigen   ' .or. branch .eq. 'EIGEN   ') go to 130
0083           if(branch .eq. 'profile ' .or. branch .eq. 'PROFILE ') go to 200
0084           if(branch .eq. 'colfreq ' .or. branch .eq. 'COLFREQ ') go to 250
0085           if(branch .eq. 'preseg  ' .or. branch .eq. 'PRESEG  ') go to 400
0086           if(branch .eq. 'coord   ' .or. branch .eq. 'COORD   ') go to 500
0087           if(branch .eq. 'quit    ' .or. branch .eq. 'QUIT    ') go to 999
0088     c
0089           print *,'ABORT MAIN:  Control card not recognized '
0090           stop
0091     c
0092     c     Path identification
0093     20    read(5,1000) pathid
0094           print 1001,pathid
0095           go to 10
0096     c
0097     c     NAMELIST input
0098     100   read(5,datum)
0099           if(nprint .gt. 1) print datum
0100           if(freq .eq. 0.) then
0101             print *,'ABORT MAIN:  FREQ not input '
0102             stop
0103           end if
0104           if(magfld .gt. 1.e-02) magfld=magfld*1.e-04
0105           go to 10
0106     c
0107     c     Separate EIGEN list input
0108     130   read(bcd,1004) fname
0109           open(unit=2,file=fname,status='old')
0110           read(2,datum)
0111           close(unit=2)
0112           if(nprint .gt. 1) then
0113             do 131 m=1,60
0114             if(eigen(1,m) .eq. 0.) go to 132
```

```
0115    131      km=m
0116    132      print 1040,d,h,(eigen(1,k),eigen(2,k),k=1,km)
0117           end if
0118           go to 10
0119    c
0120    c      Profile input
0121    200    read(bcd,1002) number
0122           nrspec=max0(1,number)
0123           nprof=0
0124           call profin(5,1,50,nprint,nrspec,lhtmx,htlist,lnlist)
0125           if(lhtmx .le. 0) then
0126             print *,'ABORT MAIN:  Ionospheric profile missing'
0127             stop
0128           end if
0129           go to 10
0130    c
0131    c      Collision frequency profile input
0132    250    nuflag=1
0133           call profin(5,2,50,nprint,nrspec,mhtmx,hclist,cflist)
0134           if(mhtmx .le. 0) then
0135             print *,'ABORT MAIN:  Collision frequency profile missing'
0136             stop
0137           end if
0138           go to 10
0139    c
0140    c      Presegmented path
0141    400    npath=2
0142           go to 600
0143    c
0144    c      Automatic path segmentation
0145    500    npath=midpnt
0146    c
0147    c      Test all inputs before execution.
0148    c
0149    c      Count the modes
0150    600    do 602 m=1,60
0151           if(eigen(1,m) .eq. 0.) go to 603
0152    602    nmds=m
0153    603    if(nmds .le. 0) then
0154             print *,'ABORT MAIN:  No EIGEN list '
0155             stop
0156           end if
0157    c      Delete duplicate modes using DEIGEN
0158           if(nmds .gt. 1) then
0159             m=1
0160             l=2
0161    605      if(abs(eigen(1,m)-eigen(1,l)) .lt. deigen(1) .and.
0162          $     abs(eigen(2,m)-eigen(2,l)) .lt. deigen(2)) then
0163    c          Found a match so drop this mode.
0164               do 607 k=l,nmds
0165               eigen(1,k)=eigen(1,k+1)
0166    607        eigen(2,k)=eigen(2,k+1)
0167               eigen(1,nmds)=0.
0168               eigen(2,nmds)=0.
0169               nmds=nmds-1
0170               if(l .le. nmds) go to 605
0171             end if
```

```
0172            if(l .lt. nmds) then
0173              l=l+1
0174              go to 605
0175            end if
0176            if(m .lt. nmds) then
0177              m=m+1
0178              l=m+1
0179              go to 605
0180            end if
0181          end if
0182    c
0183    610   if(nmds .gt. 30) then
0184    c       Too many modes input, reduce the number by deleting input
0185    c       eigen list values which have attenuation rates in excess
0186    c       of atnmax and re-count the modes
0187            capk=1./(1.-.5*alpha*h)
0188            aconst=-182.0426*freq
0189            atnmx=atnmax
0190    611     nm=0
0191            do 614 m=1,nmds
0192            if(eigen(1,m) .eq. 0.) go to 615
0193            theta=cmplx(eigen(1,m),eigen(2,m))*(.01745329252,0.)
0194            if(aconst*aimag(capk*csin(theta)) .le. atnmx) then
0195              if(nm .eq. 30) then
0196                antmx=atnmx-5.
0197                go to 611
0198              else
0199                nm=nm+1
0200                elist(1,nm)=eigen(1,m)
0201                elist(2,nm)=eigen(2,m)
0202              end if
0203            end if
0204    614     continue
0205    615     nmds=nm
0206            if(nprint .gt. 1)
0207         $    print 1042,atnmax,(elist(1,k),elist(2,k),k=1,nmds)
0208          else
0209    c       Keep all input modes.
0210            do 616 m=1,nmds
0211            elist(1,m)=eigen(1,m)
0212    616     elist(2,m)=eigen(2,m)
0213          end if
0214          if(nmds .lt. 30) then
0215            elist(1,nmds+1)=0.
0216            elist(2,nmds+1)=0.
0217          end if
0218    c
0219          if(rpoly .eq. 1 .and. tlist(1,1) .eq. 0.) then
0220            do 619 m=1,nmds
0221            tlist(1,m)=elist(1,m)
0222    619     tlist(2,m)=elist(2,m)
0223          end if
0224    c
0225          if(nuflag .eq. 0) then
0226            mhtmx=2
0227            hclist(1)=200.
0228            hclist(2)=0.
```

```
0229            do 641 n=1,nrspec
0230            en=alog(coefnu(n))
0231            cflist(1,n)=en+expnu(n)*hclist(1)
0232    641     cflist(2,n)=en+expnu(n)*hclist(2)
0233          end if
0234    c
0235          if(nprof .eq. 1) then
0236            if(beta*hprime .eq. 0. .and. npath .ne. 2) then
0237    c         This is not a presegmented path, the profile specification
0238    c         must be made in the NAMELIST.
0239              print *,'ABORT MAIN:  BETA or HPRIME not input '
0240              stop
0241            end if
0242            nrspec=1
0243            lhtmx=2
0244            htlist(1)=200.
0245            htlist(2)=0.
0246            hprout=hprime
0247          else
0248            if(nprof .eq. 2) then
0249    c         Non-exponential profile, get a value for HPRIME
0250              call gethpr(wr0,hprout)
0251            end if
0252          end if
0253    c
0254    c     BEGIN:
0255    c
0256          call gcpath
0257          go to 10
0258    c
0259    999   stop
0260    1000  format(a)
0261    1001  format(1x,(a))
0262    1002  format(8x,i1)
0263    1004  format(8x,a)
0264    1040  format(' Input   EIGEN list:  D=',f5.2,' H=',f5.2/
0265         $       ' EIGEN =',6(f8.3,' ')/(8x,6(f8.3,' ')))
0266    1042  format(' Reduced EIGEN list:  ATNMAX=',f5.1/
0267         $       ' EIGEN =',6(f8.3,' ')/(8x,6(f8.3,' ')))
0268          end
```

```
0001          function cdang(arg)
0002          complex*16 arg
0003          real*8 cdang,argr,argi
0004          argr=dreal(arg)
0005          argi=dimag(arg)
0006          cdang=datan2(argi,argr)
0007          if(argi .ge. 0.d0) return
0008          cdang=cdang+6.2831853072d0
0009          return
0010          end
```

```
0001          subroutine comp f
0002          implicit real*8 (a-h,o-z)
0003      c
0004          include 'common2.for'
0018          include 'common3.for'
0043      c
0044          c=cdcos(theta*zdtr)
0045          csq=c*c
0046          s=cdsin(theta*zdtr)
0047          ssq=s*s
0048          call rbars
0049          if(rpoly .eq. 0) then
0050            call integ
0051          else
0052            call uspoly
0053          end if
0054          if(typitr-1) 5,10,15
0055      5   f=(rbar11*r11-zone)*(rbar22*r22-zone)
0056         $  -rbar11*rbar22*r12*r21
0057          return
0058      10  f=rbar11*r11-zone
0059          return
0060      15  f=rbar22*r22-zone
0061          return
0062          end
```

```
0001          subroutine drvequ
0002          implicit real *8 (a-h,o-z)
0003      c
0004          include 'common1.for'
0020          include 'common2.for'
0034          include 'common3.for'
0059      c
0060          complex*16 k2i,il,im,in,capd,usqd,yud,ysqd,u,usq,
0061         $           t11,t31,t42,t44,t12vrc,t14vrc,t32vrc,t34vrc,ct41,
0062         $           s11a,d11a,s11b,d11b,c12,c21,
0063         $           s12,d12,s21,d21,s22,d22,b11,b22,b12,b21
0064          real*8 lsq,msq,nsq,lm,ln,mn
0065          real*4 ht0
0066          dimension cx(3),capy(3),ysq(3)
0067          data dtr/1.745329252d-2/,coeffy/1.758796d11/,coeffx/3.182357d09/
0068      c
0069          entry intcmp
0070          k2i=dcmplx(0.d0,-0.5d0*wn)
0071          sindip=dsin(codip*dtr)
0072          drcosl=sindip*dcos(azim*dtr)
0073          drcosm=sindip*dsin(azim*dtr)
0074          drcosn=-dcos(codip*dtr)
0075          il=dcmplx(0.d0,drcosl)
0076          im=dcmplx(0.d0,drcosm)
0077          in=dcmplx(0.d0,drcosn)
0078          lsq=drcosl**2
0079          msq=drcosm**2
0080          nsq=drcosn**2
0081          lm=drcosl*drcosm
0082          ln=drcosl*drcosn
0083          mn=drcosm*drcosn
0084          c0=coeffx/omega**2
0085          cy=coeffy*magfld/omega
0086          do 1 k=1,nrspec
0087          cx(k)=c0*charge(k)**2/mratio(k)
0088          capy(k)=cy*charge(k)/mratio(k)
0089      1   ysq(k)=capy(k)**2
0090          call gethpr(100.*wr0,ht0)
0091          topht=ht0
0092          lhtmn=lht
0093          mhtmn=mht
0094          if(debug .le. 1) return
0095          print 110
0096          l=lhtmn
0097          m=mhtmn
0098          ht=topht
0099      10  slopel=(ht-htlist(l+1))/(htlist(l)-htlist(l+1))
0100          slopem=(ht-hclist(m+1))/(hclist(m)-hclist(m+1))
0101          ed=dexp(lnlist(l+1,1)+(lnlist(l,1)-lnlist(l+1,1))*slopel)
0102          en=dexp(cflist(m+1,1)+(cflist(m,1)-cflist(m+1,1))*slopem)
0103          capx=ed*cx(1)
0104          capz=en/omega
0105          wr=omega*capx/capz
0106          print 111,ht,ed,en,capx,capz,wr
0107          if(ht .lt. topht) return
0108          ht=d
0109          do 11 j=l,lhtmx
```

```
0110            if(d .ge. htlist(j)) go to 12
0111     11     l=j
0112     12     do 13 j=m,mhtmx
0113            if(d .ge. hclist(j)) go to 10
0114     13     m=j
0115     c
0116            entry smtrix
0117            usqd=zero
0118            yud=zero
0119            ysqd=zero
0120            slopel=(ht-htlist(lht+1))/(htlist(lht)-htlist(lht+1))
0121            slopem=(ht-hclist(mht+1))/(hclist(mht)-hclist(mht+1))
0122            do 20 k=1,nrspec
0123            capx=dexp(lnlist(lht+1,k)
0124           $        +(lnlist(lht,k)-lnlist(lht+1,k))*slopel)*cx(k)
0125            capz=dexp(cflist(mht+1,k)
0126           $        +(cflist(mht,k)-cflist(mht+1,k))*slopem)/omega
0127            u=dcmplx(1.d0,-capz)
0128            usq=u*u
0129            capd=-capx/(u*(usq-ysq(k)))
0130            if(cdabs(capd) .gt. 1.d-30) then
0131              usqd=usqd+usq*capd
0132              yud=yud+capy(k)*u*capd
0133              ysqd=ysqd+ysq(k)*capd
0134            end if
0135     20     continue
0136            crvtrm=alpha*(h-ht)
0137            m11=usqd-lsq*ysqd-crvtrm
0138            m22=usqd-msq*ysqd-crvtrm
0139            m33=usqd-nsq*ysqd-crvtrm
0140            m12=-in*yud-lm*ysqd
0141            m21= in*yud-lm*ysqd
0142            m13= im*yud-ln*ysqd
0143            m31=-im*yud-ln*ysqd
0144            m23=-il*yud-mn*ysqd
0145            m32= il*yud-mn*ysqd
0146            capd=zone/(zone+m33)
0147            t11=-s*m31*capd
0148            t12vrc=s*m32*capd/c
0149            t14vrc=(csq+m33)*capd/c
0150            t31=m23*m31*capd-m21
0151            t32vrc=c+(m22-m23*m32*capd)/c
0152            t34vrc=s*m23*capd/c
0153            ct41=(zone+m11-m13*m31*capd)*c
0154            t42=m32*m13*capd-m12
0155            t44=-s*m13*capd
0156            s11a=t11+t44
0157            d11a=t11-t44
0158            s11b=t14vrc+ct41
0159            d11b=t14vrc-ct41
0160            s12=t12vrc+t42
0161            d12=t12vrc-t42
0162            s21=t34vrc+t31
0163            d21=t34vrc-t31
0164            s22=c+t32vrc
0165            d22=c-t32vrc
0166     c
```

DRVEQU

```
0167          if(ht .eq. topht) call intalr
0168    c
0169          entry rderiv
0170          k=0
0171          do 30 j=1,7,2
0172          k=k+1
0173          if(dabs(logr(j)) .gt. 15.d0)
0174         $  logrs(k)=dcmplx(dsign(15.d0,logr(j)),0.d0)
0175    30    rs(k)=cdexp(logrs(k))
0176          b11=r11*(d11a-d11b)
0177          b22=r22*d22
0178          b12=r12*d21
0179          b21=r21*s12
0180          c12=r12*s21
0181          c21=r21*d12
0182          d111dh=k2i*
0183         $       (b11+b12+b21-s11b-s11b+(r12*r21*d22+c12+c21-d11a-d11b)/r11)
0184          d122dh=k2i*
0185         $       (b12+b21+b22-s22-s22+(r12*r21*(d11a-d11b)+b12+b21+d22)/r22)
0186          d112dh=k2i*
0187         $       (b11+b12+b22+s11a-s11b-s22+(r11*s12+d12)*(r22+zone)/r12)
0188          d121dh=k2i*
0189         $       (b11+b21+b22-s11a-s11b-s22+(r11*d21+s21)*(r22+zone)/r21)
0190    c
0191          if(debug .gt. 2) then
0192            print 100,ht,delh,logr,dlrdh
0193          end if
0194          return
0195    c
0196    100   format(f9.4,1pe12.4,4(1x,2e12.3)/21x,4(1x,2e12.3))
0197    110   format(/' Electron density parameters:  ht    den       nu',
0198         $        8x,'x         z         w')
0199    111   format(27x,f7.1,1p5e10.2)
0200          end
```

```
0001          subroutine extrap
0002      c
0003      c     This routine sets up and maintains the data sets for the quadratic
0004      c     extrapolation of eigen's down the propagation path.
0005      c
0006          include 'common1.for'
0022          include 'common2.for'
0036      c
0037          logical brwstr
0038          complex*8 t(30),y(30),ys(3,30),s,tb,stb,capk,coeff,ngsq,
0039         $          zero/(0.,0.)/,zone/(1.,0.)/,zmplxi/(0.,1.)/
0040          dimension xs(3)
0041          equivalence (elist,y),(tlist,t)
0042          data dtr/.01745329252/
0043      c
0044          if(lx .eq. 0) then
0045      c     This is the first point on the propagation path.
0046      c     Set up constants and remove input modes with attenuation
0047      c     rates greater than atnmax.
0048            capk=cmplx(1.-.5*alpha*h,0.)
0049            coeff=cmplx(0.,182.0428*freq)/capk
0050      c
0051      c     Get Brewster mode
0052            if(sigma .lt. 1.e-3) then
0053              ngsq=cmplx(epsr,-1.7975e7*sigma/freq)
0054              stb=csqrt(ngsq/(ngsq+zone))*capk
0055              atten=coeff*stb
0056              if(atten .le. atnmax) then
0057      c         The Brewster mode is contained within the normal set.
0058                tb=(90.,0.)
0059                brwstr=.false.
0060              else
0061      c         The Brewster mode is outside the normal set.
0062                if(atten .le. 2.*atnmax) then
0063      c           The attenuation rate is not excessive.
0064                  tb=cmplx(0.,-1./dtr)*clog(csqrt(zone-stb**2)+zmplxi*stb)
0065                  brwstr=.true.
0066                else
0067      c           The attenuation rate is excessive.
0068                  tb=(90.,0.)
0069                  brwstr=.false.
0070                end if
0071              end if
0072            else
0073              tb=(90.,0.)
0074              brwstr=.false.
0075            end if
0076            do 139 k=1,30
0077            mode(k)=k
0078      137   if(real(y(k)) .gt. 0.) then
0079              if(brwstr) then
0080      c         If this mode is near the Brewster, then keep it.
0081                if(abs( real(y(k)-tb)) .le. 1. .and.
0082           $       abs(aimag(y(k)-tb)) .le. .5) go to 139
0083              end if
0084              atten=coeff*csin(y(k)*dtr)
0085              if(atten .gt. atnmax) then
```

```
0086                do 138 l=k,30
0087                t(l)=t(l+1)
0088    138         y(l)=y(l+1)
0089                t(30)=zero
0090                y(30)=zero
0091                go to 137
0092              end if
0093            end if
0094    139     continue
0095            return
0096          end if
0097    c
0098          x=rho
0099          if(nprint .gt. 1) print 1000,x
0100          nmds=ls
0101          do 143 k=1,nmds
0102          s=zero
0103          do 142 l1=1,lx
0104          p=1.
0105          do 141 l2=1,lx
0106          if(l1 .eq. l2) go to 141
0107          p=p*(x-xs(l2))/(xs(l1)-xs(l2))
0108    141   continue
0109    142   s=s+p*ys(l1,k)
0110          if(nprint .gt. 1) print 1001,mode(k),s
0111          t(k)=s
0112    143   y(k)=s
0113    c
0114    c     Scan the extrapolated eigen's for invalid values.
0115          do 151 k=1,nmds
0116          if(real(y(k)) .le. 0. .or. real(y(k)) .ge. 90. .or.
0117         $  aimag(y(k)) .ge. 0.) then
0118            print *,'ERROR EXTRAP:  Extrapolated mode',mode(k)
0119            lost=1
0120            return
0121          end if
0122    151   continue
0123          if(nmds .lt. 30) then
0124            t(nmds+1)=zero
0125            y(nmds+1)=zero
0126          end if
0127          return
0128    c
0129          entry xsave
0130    c
0131    c     This entry point is called after execution of WVGUID.
0132    c     It updates the data sets used to do the quadratic extrapolation.
0133    c
0134          x=rho
0135          if(lx .lt. 3) then
0136            lx=lx+1
0137          else
0138            do 21 l=1,2
0139            xs(l)=xs(l+1)
0140            do 21 k=1,nmds
0141    21      ys(l,k)=ys(l+1,k)
0142          end if
```

```
0143          ls=nmds
0144          xs(lx)=x
0145          do 25 k=1,nmds
0146    25    ys(lx,k)=y(k)
0147    c
0148    c     Keep all eigen's at first input distance.
0149          if(lx .eq. 1) then
0150            modes=nmds
0151            return
0152          end if
0153    c     j is counter for modes to be output by SAVEMC
0154    c     k is counter for modes to be used by WVGUID
0155          j=0
0156          k=1
0157    251   if(k .gt. nmds) return
0158          j=j+1
0159          if(brwstr) then
0160    c       Check if this mode is near the Brewster; if so, then keep it.
0161            if(abs( real(y(k)-tb)) .le. 1. .and.
0162         $     abs(aimag(y(k)-tb)) .le. .5) go to 256
0163          end if
0164          atten=coeff*csin(y(k)*dtr)
0165          if(atten .gt. atnmax) then
0166    c       Delete k-th mode
0167            do 253 l=1,4
0168    253     T term(l,j)=zero
0169            nmds=nmds-1
0170            ls=nmds
0171            if(nmds .eq. 0) then
0172              print *,'ERROR EXTRAP:  All modes have been deleted'
0173              lost=1
0174              t(1)=zero
0175              y(1)=zero
0176              return
0177            end if
0178            if(k .gt. nmds) then
0179    257       modes=modes-1
0180              if(real(T term(1,modes)) .ne. 0.) return
0181              go to 257
0182            else
0183              do 254 l=k,nmds
0184              mode(l)=mode(l+1)
0185              t(l)=t(l+1)
0186              y(l)=y(l+1)
0187              do 254 m=1,lx
0188              ys(m,l)=ys(m,l+1)
0189    254       continue
0190              t(nmds+1)=zero
0191              y(nmds+1)=zero
0192              go to 251
0193            end if
0194          end if
0195    256   k=k+1
0196          go to 251
0197    c
0198    1000  format(/' Extrapolated EIGEN list for x =',f8.3)
0199    1001  format(i5,5x,2f8.3,f12.3)
0200          end
```

```
0001            subroutine gcdbr(dl,clt1,clt2,rho,br,inb)
0002    c
0003    c       Returns great circle distance and geographic bearing angle
0004    c
0005    c       Input:  DL is longitude of point 2 minus longitude of point 1
0006    c               CLT1 is co-latitude of point 1
0007    c               CLT2 is co-latitude of point 2
0008    c               INB=0:  RHO is computed and
0009    c                       BR from point 1 thru point 2 is computed at 1
0010    c               INB=1:  RHO is input    and
0011    c                       BR from point 1 thru point 2 is computed at 2
0012    c
0013    c       Output:  RHO is great circle distance between the input points
0014    c                BR  is geographic bearing angle measured clockwise from
0015    c                       due North
0016    c
0017    c       All coordinates, RHO and BR are in radians
0018    c       Sign convention is + for West and North
0019    c
0020            data pi/3.14159265e0/,twopi/6.28318531e0/
0021    c
0022            reduce(arg)=sign(amin1(abs(arg),1.),arg)
0023    c
0024            cclt1=cos(clt1)
0025            sclt1=sin(clt1)
0026            cclt2=cos(clt2)
0027            sclt2=sin(clt2)
0028    c
0029            adl=abs(dl)
0030            if(adl .ge. pi) then
0031              dl=amod(dl,twopi)
0032            else
0033              adl=abs(dl)
0034            end if
0035            if(inb .eq. 1) then
0036              if(rho .gt. pi) then
0037                gcd=twopi-rho
0038              else
0039                gcd=rho
0040              end if
0041            end if
0042            if(abs(clt1) .le. 1.e-6 .or. abs(clt1-pi) .le. 1.e-6) go to 10
0043            if(adl .le. 1.e-6) go to 20
0044            if(abs(adl-pi) .le. 1.e-6) go to 30
0045            if(adl .ge. pi) then
0046              if(dl .ge. 0.) then
0047                dl=dl-twopi
0048              else
0049                dl=dl+twopi
0050              end if
0051            end if
0052            if(inb .eq. 0) then
0053              cgcd=cclt1*cclt2+sclt1*sclt2*cos(dl)
0054              gcd=acos(reduce(cgcd))
0055              if(abs(cgcd-1.) .le. 1.e-6) then
0056                br=0.
0057              else
```

```
0058                br=acos(reduce((cclt2-cclt1*cgcd)/(sclt1*sin(gcd))))
0059              end if
0060            else
0061              if(abs(gcd) .le. 1.e-6) then
0062                br=0.
0063              else
0064                br=pi-acos(reduce((cclt1-cclt2*cos(gcd))/(sclt2*sin(gcd))))
0065              end if
0066            end if
0067            if(dl .lt. 0.) br=twopi-br
0068            go to 40
0069    c
0070    c       point 1 is at one of the poles
0071    10      if(inb .eq. 0) gcd=abs(clt1-clt2)
0072            if(abs(clt1) .le. 1.e-6) then
0073              br=pi-dl
0074            else
0075              br=dl
0076            end if
0077            go to 40
0078    c
0079    c       coordinates are on same longitude
0080    20      dc=clt1-clt2
0081            if(dc .ge. 0.) then
0082              br=0.
0083            else
0084              dc=-dc
0085              br=pi
0086            end if
0087            if(inb .eq. 0) gcd=dc
0088            go to 40
0089    c
0090    c       coordinates are on opposite longitudes
0091    30      dc=clt1+clt2
0092            if(dc .le. pi) then
0093              if(inb .eq. 0) then
0094                br=0.
0095              else
0096                br=pi
0097              end if
0098            else
0099              dc=twopi-dc
0100              if(inb .eq. 0) then
0101                br=pi
0102              else
0103                br=0.
0104              end if
0105            end if
0106            if(inb .eq. 0) gcd=dc
0107    c
0108    c       long path calculations
0109    40      if(inb .eq. 1) then
0110              if(rho .gt. pi) then
0111                if(br .lt. pi) then
0112                  br=br+pi
0113                else
0114                  br=br-pi
```

```
0115                end if
0116              end if
0117            end if
0118    c
0119    90      if(inb .eq. 0) rho=gcd
0120            return
0121            end
```

```
0001            subroutine gcpath
0002      c
0003      c     sign convention: + for west and north, - for east and south
0004      c
0005            include 'common1.for'
0021            include 'common2.for'
0035      c
0036            dimension prof1(50,3),prof2(50,3)
0037            real lng,long,lat,m1,m2,mgf
0038            character*72 bcd,preseg
0039            logical first
0040            data dtr/1.745329e-2/,re/6.366/,alt/80./
0041      c
0042      c     min=0 --- normal
0043      c        =1 --- interpolating between preseg values
0044      c        =2 --- last interpolation interval
0045      c
0046      c     lost=0 -- no trouble with modes
0047      c         =1 -- dropped a mode in WVGUID or EXTRAP
0048      c         =2 -- all modes found but one or more changed significantly
0049      c               from the extrapolated values
0050      c
0051      c     nprof=0 - use profile from MAIN
0052      c           1 - use exponential profile
0053      c           2 - read non-exponential profiles along path
0054      c               WARNING:  the heights must match in each profile
0055      c
0056            first=.true.
0057            write(90,*) 'Failure at RHO 1'
0058      c
0059            lx=0
0060            min=0
0061            bta=0.
0062            sig=0.
0063            mgf=0.
0064            sigma1=0.
0065            tlng=tlong*dtr
0066            tclt=(90.-tlat)*dtr
0067            rho0=rho
0068            rhop=rho
0069            drho=drmin
0070            if(rbear .eq. 720.) then
0071              call gcdbr((tlong-rlong)*dtr,tclt,(90.-rlat)*dtr,gcd,xtr,0)
0072              brng=xtr/dtr
0073              if(igcd .eq. 1) then
0074                rhomax=gcd*re
0075              else
0076                rhomax=dmax
0077              end if
0078            else
0079              xtr=rbear*dtr
0080              brng=rbear
0081              rhomax=dmax
0082            end if
0083      c
0084      20    if(npath .eq. 2) then
0085      c       Presegmented path
```

```
0086            read(5,2000,end=900) preseg
0087            read(preseg,*) rho,azm,cdp,mgf,sig,eps,bta,hprm
0088            if(rho .eq. 40.) then
0089              print *,'End of preseg data'
0090              rewind 90
0091              write(90,2003)
0092              go to 999
0093            else if(rho .gt. rhomax) then
0094              print *,'DMAX reached before end of preseg data '
0095              rewind 90
0096              write(90,2003)
0097              go to 900
0098            end if
0099            if(first) then
0100              rho0=rho
0101              rhop=rho
0102            end if
0103            drho=rho-rhop
0104            if(drho .lt. 0.) then
0105              print *,'ABORT GCPATH:  Preseg rhos out of order'
0106              go to 900
0107            end if
0108            if(nprof .eq. 2) then
0109              read(3,2000) bcd
0110              if(bcd(1:8) .ne. 'profile ' .and.
0111         $       bcd(1:8) .ne. 'PROFILE ') then
0112                print *,'ABORT GCPATH:  PROFILE control string missing'
0113                go to 900
0114              end if
0115              read(bcd,2001) nn
0116              if(nrspec .ne. max0(1,nn)) then
0117                print *,'ABORT GCPATH:  Number of species is incorrect'
0118                go to 900
0119              end if
0120              call profin(3,1,50,nprint,nrspec,lhtmx,htlist,lnlist)
0121              if(lhtmx .lt. 0) then
0122                print *,'ABORT GCPATH:  Profile missing'
0123                go to 900
0124              end if
0125              if(lhtmx .gt. 0) then
0126                if(lhtmx .ne. lhtmx1) then
0127                  print *,'ABORT GCPATH:  Number of heights is incorrect'
0128                  go to 900
0129                end if
0130                call gethpr(wr0,hprout)
0131              end if
0132            else
0133              if(nprof .eq. 1) then
0134                if(bta .gt. 0.) then
0135                  beta=bta
0136                  hprime=hprm
0137                end if
0138                if(beta*hprime .eq. 0.) then
0139                  print *,'ABORT GCPATH:  BETA or HPRIME not input'
0140                  go to 900
0141                end if
0142              end if
```

```
0143    c         Calculate exponential profile:
0144              hnlist(1,1)=cflist(1,1)+beta*(htlist(1)-hprime)-9.4517306
0145              hnlist(2,1)=cflist(2,1)+beta*(htlist(2)-hprime)-9.4517306
0146              hprout=hprime
0147            end if
0148          end if
0149    c
0150          if(npath .eq. 1) then
0151    c       Calculate midpoint distance:
0152            rho=.5*rhomax
0153          else
0154            if(rho .eq. 0.) then
0155    c         Begin at xmtr
0156              lng=tlng
0157              clt=tclt
0158              br=xtr
0159            end if
0160          end if
0161    c
0162    30    if(rho .gt. 0.) then
0163            gcd=rho/re
0164            call recvr(tlng,tclt,xtr,gcd,lng,clt)
0165            if(rho .eq. 20.) then
0166    c         At the antipode of the transmitter.
0167              br=9.4247779608-xtr
0168              if(br .gt. 6.2831853072) br=br-6.2831853072
0169            else
0170              call gcdbr(tlng-lng,tclt,clt,gcd,br,1)
0171            end if
0172          end if
0173          bpath=br/dtr
0174          long=lng/dtr
0175          colat=clt/dtr
0176          lat=90.-colat
0177    c
0178          if(sig .gt. 0.) then
0179            sigma=sig
0180            epsr=eps
0181          else
0182            if(ignd .eq. 1) call ground(long,lat,ncode,sigma,epsr)
0183          end if
0184    c     If conductivity has changed, then restart extrapolation
0185          if(sigma1 .ne. sigma .and. sigma1 .ne. 0.) then
0186            min=0
0187            lx=0
0188            call xsave
0189            lx=0
0190          end if
0191    c
0192          if(mgf .gt. 0.) then
0193            azim=azm
0194            codip=cdp
0195            magfld=mgf
0196            if(magfld .gt. 1.e-02) magfld=magfld*1.e-04
0197          else
0198            call newmag(0,alt,lng,clt,bmf,dip,b,br,bp,bt)
0199            azim=bpath-bmf/dtr
```

```
0200            if(azim .lt. 0.) then
0201              azim=azim+360.
0202            else if(azim .gt. 360.) then
0203              azim=azim-360.
0204            end if
0205            codip=90.-dip/dtr
0206            magfld=b*1.0e-04
0207    c
0208            if(mdir .eq. 1) then
0209    c         Reverse azim if contours for xmtr deployment
0210              azim=azim-180.
0211              if(azim .lt. 0.) then
0212                azim=azim+360.
0213              else if(azim .gt. 360.) then
0214                azim=azim-360.
0215              end if
0216            end if
0217    c
0218          end if
0219          print 1000,rho,long,lat,bpath,azim,codip,magfld,sigma,epsr
0220    c
0221    40    lost=0
0222          x=rho
0223          call extrap
0224          if(lost .eq. 1) go to 100
0225          call wvguid
0226          if(nmds .eq. 0 .and. (rho .eq. 0. .or. npath .eq. 1 )) then
0227            print *,'ABORT GCPATH:  Failure at starting rho'
0228            go to 900
0229          end if
0230          if(lost .eq. 1) go to 100
0231          call xsave
0232          if(lost .eq. 1) go to 100
0233          rhop=rho
0234          if(min .eq. 1) go to 50
0235          if(first) then
0236    c       Primary output file:
0237            open(unit=lunit7,status='new')
0238            if(year .eq. 0 .and. month .eq. 0 .and. day .eq. 0) then
0239              write(lunit7,1030) tlong,tlat,brng,beta,hprime,pathid
0240            else
0241              write(lunit7,1031) tlong,tlat,brng,beta,hprime,
0242         $                       mod(year,100),month,day,gmt,pathid
0243            end if
0244            first=.false.
0245          end if
0246          call savemc
0247          if(npath .eq. 1) then
0248            rewind 90
0249            write(90,2003)
0250            go to 999
0251          else
0252            rewind 90
0253            write(90,2002) rho
0254          end if
0255          if(rho+.002 .ge. rhomax) go to 900
0256    c
```

```
0257            rho1=rho
0258            a1=azim
0259            c1=codip
0260            m1=magfld
0261            e1=epsr
0262            s1=alog(sigma)
0263            sigma1=sigma
0264            if(nprof .gt. 0) then
0265              lhtmx1=lhtmx
0266              do 48 l=1,lhtmx
0267              do 48 m=1,nrspec
0268     48       prof1(l,m)=lnlist(l,m)
0269            end if
0270            if(min .eq. 2) then
0271              min=0
0272              go to 70
0273            end if
0274     c
0275     50     if(lost .eq. 2) then
0276              if(min .eq. 0) then
0277                drho=amax1(drho-drmin,drmin)
0278              else
0279                drho=.5*(rho2-rho)
0280              end if
0281            else
0282              if(min .eq. 0) then
0283                drho=amin1(drho+drmin,drmax)
0284              else
0285                drho=rho2-rho
0286              end if
0287            end if
0288     c
0289     70     if(min .eq. 0 .and. npath .eq. 2) go to 20
0290            rho=rho+drho
0291            if(rho+.002 .gt. rhomax) then
0292              drho=drho-rho+rhomax
0293              rho=rhomax
0294            end if
0295            if(min .eq. 1) go to 120
0296            go to 30
0297     c
0298     c      Back up on propagation path
0299     100    if(rho .eq. rho0) then
0300              print *,'ABORT GCPATH:  Failure at starting rho'
0301              go to 900
0302            end if
0303            if(min-1) 101,102,103
0304     101    if(npath .eq. 2) go to 105
0305            if(drho .le. drmin) go to 110
0306            nrd=.5*drho/drmin
0307            if(nrd .eq. 0) go to 110
0308            drho=nrd*drmin
0309            rho=rho1+drho
0310            go to 30
0311     103    min=1
0312            drho=drhop
0313     102    drho=.5*drho
```

```
0314          if(drho .lt. .015125) then
0315            print *,'ABORT GCPATH:  Backup interval is less than 0.015125'
0316            go to 900
0317          end if
0318    104   rho=rhop+drho
0319          go to 120
0320    c
0321    105   if((rho-rho1)/drmin .gt. 10.) then
0322            print *,'ABORT GCPATH:  Preseg interval too large for efficient
0323         $processing'
0324            go to 900
0325          end if
0326    c
0327    c     Begin interpolation
0328    110   min=1
0329          drho=.5*(rho-rho1)
0330          if(drho .lt. .015125) then
0331            print *,'ABORT GCPATH:  Backup interval is less than 0.015125'
0332            go to 900
0333          end if
0334          rho2=rho
0335          a2=azim
0336          if(a2-a1 .gt.  180.) then
0337            a2=a2-360.
0338          else if(a2-a1 .lt. -180.) then
0339            a2=a2+360.
0340          end if
0341          c2=codip
0342          m2=magfld
0343          e2=epsr
0344          s2=alog(sigma)
0345          sigma2=sigma
0346          if(nprof .gt. 0) then
0347            lhtmx2=lhtmx
0348            do 111 l=1,lhtmx
0349            do 111 m=1,nrspec
0350    111     prof2(l,m)=lnlist(l,m)
0351          end if
0352          rho=rho1+drho
0353    c
0354    120   if(rho+.002 .ge. rho2) then
0355    c       End of interpolation
0356            min=2
0357            drhop=drho
0358            drho=drmin
0359            rho=rho2
0360            azim=a2
0361            if(azim .lt. 0.) then
0362              azim=azim+360.
0363            else if(azim .gt. 360.) then
0364              azim=azim-360.
0365            end if
0366            codip=c2
0367            magfld=m2
0368            epsr=e2
0369            sigma=sigma2
0370            if(nprof .gt. 0) then
```

```
0371              do 121 l=1,lhtmx
0372              do 121 m=1,nrspec
0373    121         lnlist(l,m)=prof2(l,m)
0374              end if
0375            else
0376    c         Interpolate
0377              slope=(rho-rho1)/(rho2-rho1)
0378              azim=a1+slope*(a2-a1)
0379              if(azim .lt. 0.) then
0380                azim=azim+360.
0381              else if(azim .gt. 360.) then
0382                azim=azim-360.
0383              end if
0384              codip=c1+slope*(c2-c1)
0385              magfld=m1+slope*(m2-m1)
0386              epsr=e1+slope*(e2-e1)
0387              sigma=exp(s1+slope*(s2-s1))
0388              if(nprof .gt. 0) then
0389                do 122 l=1,lhtmx
0390                do 122 m=1,nrspec
0391    122         lnlist(l,m)=prof1(l,m)+slope*(prof2(l,m)-prof1(l,m))
0392              end if
0393            end if
0394            print 1002,   rho,azim,codip,magfld,sigma,epsr
0395            go to 40
0396    c
0397    900     if(npath .eq. 2) then
0398    903       read(5,2000,end=999) bcd
0399              if(bcd(1:5) .eq. '40,0,') go to 999
0400              go to 903
0401            end if
0402    999     write(lunit7,1032)
0403            close(unit=lunit7)
0404            print *,'Execution terminating for this path'
0405            return
0406    c
0407    1000    format(/' Propagation path parameters:   rho      long      lat',
0408           $       4x,'bear     azim     codip    magfld     sigma      epsr'/
0409           $       26x,f10.3,f10.2,4f9.2,e11.2,1pe11.2,0pf8.2)
0410    1002    format(/11x,' Interpolated path parameters:   rho      azim',
0411           $       5x,'codip    magfld     sigma      epsr'/
0412           $       38x,f10.3,f10.2,f9.2,e11.2,1pe11.2,0pf8.2)
0413    1030    format('sw   xmtr',f7.1,2f6.1,' prof',f5.2,f5.1/a80)
0414    1031    format('sw   xmtr',f7.1,2f6.1,' prof',f5.2,f5.1,
0415           $       '  (',3(i2.2,'/'),f4.1,')'/a80)
0416    1032    format('r 40.')
0417    2000    format(a72)
0418    2001    format(8x,i1)
0419    2002    format(f6.3)
0420    2003    format('40')
0421            end
```

```
0001            subroutine gethpr(wr,hpr)
0002      c
0003      c     Routine to determine the height where omega sub r is a
0004      c     specific value.  The value returned is to nearest km.
0005      c
0006            include 'common1.for'
0022      c
0023            data coeffx/3.182357e9/
0024      c
0025      c     Start at the bottom of the profile and work up.
0026      c
0027            lht=lhtmx-1
0028            mht=mhtmx-1
0029            ht=amin1(htlist(lhtmx),hclist(mhtmx))
0030      10    if(lht .gt. 1 .and. ht .ge. htlist(lht-1)) then
0031              lht=lht-1
0032              go to 10
0033            end if
0034      12    if(mht .gt. 1 .and. ht .ge. hclist(mht-1)) then
0035              mht=mht-1
0036              go to 12
0037            end if
0038            slope l=(ht-htlist(lht+1))/(htlist(lht)-htlist(lht+1))
0039            slope m=(ht-hclist(mht+1))/(hclist(mht)-hclist(mht+1))
0040            sum=0.
0041            do 14 n=1,nrspec
0042            dn=exp(lnlist(lht+1,n)+(lnlist(lht,n)-lnlist(lht+1,n))*slope l)
0043            cf=exp(cflist(mht+1,n)+(cflist(mht,n)-cflist(mht+1,n))*slope m)
0044      14    sum=sum+coeffx*dn/(mratio(n)*cf)
0045            if(sum .gt. wr) then
0046              hpr=ht
0047              if(nprint .gt. 1) then
0048                print *
0049                print *,'GETHPR:  wr, hpr=',wr,hpr
0050              end if
0051              return
0052            end if
0053            ht=ht+1.
0054            go to 10
0055            end
```

```
0001            subroutine ground(xlong,xlat,ncode,sigma,epsr)
0002     c
0003     c      Returns conductivity code, conductivity, dielectric constant
0004     c
0005     c      Input:  XLONG is West longitude in degrees
0006     c              (i.e., -117.3 for 117 degrees, 18 minutes East)
0007     c              XLAT is North latitude  in degrees
0008     c             (i.e., 32.8 for 32 degrees, 48 minutes North)
0009     c
0010     c      Output:  NCODE is conductivity code from GRNDMAP.DAT
0011     c               (ncode=0 is sea water, =1 is ice; see DATA below)
0012     c               SIGMA is mho/m
0013     c               EPSR is the dielectric constant
0014     c               A list of sigma and epsr is also placed into a common.
0015     c
0016     c      Requires:  GRNDMAP.DAT
0017     c
0018            include '[305021.jaflib]data_files.for/list'
0019   1        character*40 grnd$d/'user$disk$3:[305021.jaflib]grndmap.dat'/
0020   1        character*40 itsn$d/'user$disk$3:[305021.jaflib]itsnoise.dat'/
0021   1        character*40 wrld$d/'user$disk$3:[305021.jaflib]world.dat'/
0022     c
0023            common/grnd$/sss(10),rrr(10)
0024            logical first/.true./
0025            dimension lcode(361),map(4530),ss(10),rr(10)
0026            data ss/1.e-5,3.e-5,1.e-4,3.e-4,1.e-3,3.e-3,1.e-2,3.e-2,.1,4./
0027            data rr/5.,5.,10.,10.,15.,15.,15.,15.,80.,81./
0028            if(first) then
0029              open(unit=8,file=grnd$d,status='old',readonly)
0030              read(8,1) lcode,map
0031     1        format(9i8)
0032              close(unit=8)
0033              do 2 l=1,10
0034              sss(l)=ss(l)
0035     2        rrr(l)=rr(l)
0036              first=.false.
0037            end if
0038            phi=xlong
0039            if(phi .gt. 180.) then
0040              phi=phi-360.
0041            else
0042              if(phi .lt. -180.) phi=phi+360.
0043            end if
0044            if(abs(phi) .gt. 180. .or. abs(xlat) .gt. 90.01) then
0045              print 11,xlong,xlat
0046     11       format(/' ********** Error in GROUND:  Xlong     Xlat'
0047          $           /26x,2f9.2)
0048              stop
0049            end if
0050            lat=181.-2.*xlat
0051            if(lat .gt. 360) lat=360
0052            long=361.-2.*phi
0053            if(long .gt. 720) long=1
0054            l1=lcode(lat)
0055            l2=lcode(lat+1)-1
0056            do 21 l=l1,l2
0057            maplm1=map(l)/10000
```

GROUND

```
0058            mapl=map(l)-10000*maplm1
0059            mlong=mapl/10
0060            if(mlong .ge. long) go to 31
0061    21      continue
0062    31      ncd=mapl-mlong*10
0063            mlong=maplm1/10
0064            if(mlong .lt. long) go to 41
0065            ncd=maplm1-mlong*10
0066    41      if(ncd .lt. 0 .or. ncd .gt. 9) then
0067              print 51,xlong,xlat,long,lat,l1,l2,mlong,ncode
0068    51        format(/' ********** Error in GROUND:  Xlong     Xlat',
0069         $             '  long   lat    l1    l2   mlong   ncode'
0070         $           /26x,2f9.2,4i6,2i8)
0071              stop
0072            end if
0073            ncode=ncd
0074            if(ncode .eq. 0) ncode=10
0075            sigma=ss(ncode)
0076            epsr= rr(ncode)
0077            return
0078            end
```

```
0001          subroutine intalr
0002          implicit real *8 (a-h,o-z)
0003    c
0004          include 'common2.for'
0018          include 'common3.for'
0043    c
0044          complex*16 q,p,t,d11,d13,d31,d33,delta,fnsq,froot,
0045         $           com1,com3,csqm22,csqm33,b3,b2,b1,b0
0046          dimension phase1(8), phase2(8), p(2), t(2), q(4)
0047          equivalence (logr11,phase1(1))
0048          data pi/3.141592653d0/,twopi/6.283185307d0/
0049    c
0050          if(isotrp-1) 10,100,102
0051    10    com1=zone*m11
0052          com3=zone*m33
0053          csqm22=csq*m22
0054          csqm33=csq*m33
0055          b3=0.25d0*s*(m13+m31)/com3
0056          b2=(-csqm33*com1+m13*m31-com3*csqm22+m23*m32)/(6.d0*com3)
0057          b1=s*(m12*m23+m32*m21-csqm22*(m13+m31))/(4.d0*com3)
0058          b0=(com1*csqm22*csqm33+m12*m23*m31+m32*m21*m13-m13*m31*csqm22
0059         $          -com1*m23*m32-m12*m21*csqm33)/com3
0060          call qartic(q,b3,b2,b1,b0,debug,newq)
0061    c
0062          do 30 n=1,2
0063          d11=zone*m11-q(n)**2
0064          d13=m13+s*q(n)
0065          d31=m31+s*q(n)
0066          d33=zone*m33-s**2
0067          delta=d11*d33-d13*d31
0068          p(n)=(-m12*d33+d13*m32)/delta
0069          t(n)=q(n)*p(n)-s*(-d11*m32+m12*d31)/delta
0070          pyntng=t(n)*dconjg(p(n))+q(n)
0071          if(pyntng .lt. 0.) print 201,theta,q(n),pyntng
0072    30    continue
0073          delta=(t(1)*c+p(1))*(c+q(2))-(t(2)*c+p(2))*(c+q(1))
0074          r11 =((t(1)*c-p(1))*(c+q(2))-(t(2)*c-p(2))*(c+q(1)))/delta
0075          r22 =((t(1)*c+p(1))*(c-q(2))-(t(2)*c+p(2))*(c-q(1)))/delta
0076          r12 =-2.d0*c*(t(1)*p(2)-t(2)*p(1))/delta
0077          r21 =-2.d0*c*(q(1)-q(2))/delta
0078    40    logr11=cdlog(r11)
0079          logr12=cdlog(r12)
0080          logr21=cdlog(r21)
0081          logr22=cdlog(r22)
0082          if(adjflg .eq. 1) then
0083            do 70 n=2,8,2
0084    50      if(phase1(n)-phase2(n) .le. pi) go to 60
0085            phase1(n)=phase1(n)-twopi
0086            go to 50
0087    60      if(phase2(n)-phase1(n) .le. pi) go to 70
0088            phase1(n)=phase1(n)+twopi
0089            go to 60
0090    70      continue
0091          end if
0092          do 90 n=2,8,2
0093    90    phase2(n)=phase1(n)
0094          if(debug .gt. 2) print 202
```

INTALR

```
0095          return
0096     c
0097     100  ir=1
0098          fnsq=zone*m11
0099          froot=cdsqrt(csq+m11)
0100          go to 106
0101     101  r11=(fnsq*c-froot)/(fnsq*c+froot)
0102          r22=(c-froot)/(c+froot)
0103          go to 105
0104     c
0105     102  ir=2
0106          fnsq=zone*m11
0107          froot=cdsqrt(csq+m11+m13**2/fnsq)
0108          go to 106
0109     103  com1=(s*froot+m13)/(s*fnsq+m13)
0110          r11=(c-com1)/(c+com1)
0111          ir=3
0112          froot=cdsqrt(csq+m22)
0113          go to 106
0114     104  r22=(c-froot)/(c+froot)
0115     105  r12=(1.d-20,0.d0)
0116          r21=(1.d-20,0.d0)
0117          go to 40
0118     c
0119     106  if(dimag(froot) .gt. 0.d0) froot=-froot
0120          if(ir-2) 101,103,104
0121     c
0122     201  format(' for theta=',f7.4,f9.4,' q=',1p2e11.3,
0123         $       ' poynting(z)=',e11.3)
0124     202  format(/4x,'ht',7x,'delh')
0125          end
```

```
0001          subroutine integ
0002          implicit real *8 (a-h,o-z)
0003    c
0004          include 'common1.for'
0020          include 'common2.for'
0034          include 'common3.for'
0059    c
0060          real*8 logr0
0061          integer sflag
0062          dimension logr0(8), dlrdh0(8), dlogr0(8), dlogr1(8), dlogr2(8)
0063          data dlhmin/1.953125d-3/,dlgrmx/1.d20/
0064    c
0065          factor=10.d0**(-prec)
0066          emax=factor*3.d0
0067          emin=factor*.3d0
0068          ht=topht
0069          lht=lhtmn
0070          mht=mhtmn
0071          delh=3.125d-2
0072          svdelh=delh
0073          if(debug .gt. 2) print 200,theta
0074          call smtrix
0075    c
0076    c     runge kutta
0077    10    sflag=0
0078          if(debug .gt. 2) print 201
0079    11    if(lht .lt. lthmx .and. ht .le. htlist(lht+1)) then
0080            lht=lht+1
0081            go to 11
0082          end if
0083    13    if(mht .lt. mthmx .and. ht .le. hclist(mht+1)) then
0084            mht=mht+1
0085            go to 13
0086          end if
0087          if(ht-delh .lt. htlist(lht+1)) then
0088            sflag=1
0089            saveht=htlist(lht+1)
0090            delh=ht-saveht
0091          end if
0092          if(ht-delh .lt. d) then
0093            sflag=1
0094            saveht=d
0095            delh=ht-saveht
0096          end if
0097          do 30 i=1,8
0098          logr0(i)=logr(i)
0099    30    dlrdh0(i)=dlrdh(i)
0100    c
0101    c     Try again
0102    40    do 50 i=1,8
0103          dlogr0(i)=-dlrdh0(i)*delh
0104    50    logr(i)=logr0(i)+0.5d0*dlogr0(i)
0105          ht=ht-0.5d0*delh
0106          call smtrix
0107          do 60 i=1,8
0108          dlrdh(i)=dsign(dmin1(dlgrmx,dabs(dlrdh(i))),dlrdh(i))
0109          dlogr1(i)=-dlrdh(i)*delh
```

```
0110    60    logr(i)=logr0(i)+0.5d0*dlogr1(i)
0111          call rderiv
0112          do 70 i=1,8
0113          dlrdh(i)=dsign(dmin1(dlgrmx,dabs(dlrdh(i))),dlrdh(i))
0114          dlogr2(i)=-dlrdh(i)*delh
0115    70    logr(i)=logr0(i)+dlogr2(i)
0116          ht=ht-0.5d0*delh
0117          call smtrix
0118          error=0.d0
0119          do 80 i=1,8
0120          dlrdh(i)=dsign(dmin1(dlgrmx,dabs(dlrdh(i))),dlrdh(i))
0121          dlogr4=((-dlrdh(i)*delh+dlogr0(i))/2.d0+dlogr1(i)+dlogr2(i))/3.d0
0122          logr(i)=logr0(i)+dlogr4
0123    80    error=error+(dlogr2(i)-dlogr4)**2
0124          error=dsqrt(error/8.d0)
0125          if(error .lt. emax .or. delh .le. dlhmin) go to 100
0126          sflag=0
0127          ht=ht+delh
0128          delh=0.5d0*delh
0129          if(delh .lt. dlhmin) delh=dlhmin
0130          go to 40
0131    100   call rderiv
0132          if(error .lt. emin) delh=2.*delh
0133          if(sflag .eq. 1) then
0134            delh=svdelh
0135            ht=saveht
0136          end if
0137          svdelh=delh
0138          if(ht .gt. d) go to 10
0139          return
0140    c
0141    200   format(/' DEBUG:  theta =',2f9.4)
0142    201   format(' ')
0143          end
```

```
0001          subroutine iterat
0002          implicit real *8 (a-h,o-z)
0003      c
0004      c     This routine drives the iteration to obtain solutions to the
0005      c     modal equation.
0006      c
0007          include 'common2.for'
0021          include 'common3.for'
0046      c
0047          complex*16 theta0,f0,dlthta
0048          real*4 absr,absi
0049      c
0050          nriter=0
0051          if(debug .gt. 1) then
0052            if(rpoly .eq. 0) then
0053              print 300
0054            else
0055              print 301
0056            end if
0057            print 302
0058          end if
0059      c     Store the starting angle
0060          theta0=theta
0061      10  theta=theta-dthta
0062          call comp f
0063          f0=f
0064          theta=theta+dthta
0065          call comp f
0066      c     Store the magnitude of the f-function for the starting angle
0067          if(nriter .eq. 0) fmag0=cdabs(f)
0068          nriter=nriter+1
0069          dfdtht=(f-f0)/dthta
0070          dlthta=-f/dfdtht
0071          if(debug .gt. 1) then
0072            fmag=cdabs(f)
0073            print 303,theta,fmag,dlthta,dfdtht
0074          end if
0075      c
0076          absr=dabs(dreal(dlthta))
0077          absi=dabs(dimag(dlthta))
0078          if(absr .gt. thtinc) dlthta=dlthta*(thtinc/absr)
0079          if(absi .gt. thtinc) dlthta=dlthta*(thtinc/absi)
0080          theta=theta+dlthta
0081          if(nriter .lt. maxitr .and.
0082         $  (absr .gt. lub(1) .or. absi .gt. lub(2))) go to 10
0083      c
0084          nriter=nriter+1
0085          f0=f
0086          call comp f
0087          dfdtht=(f-f0)/dlthta
0088          if(debug .gt. 1) then
0089            fmag=cdabs(f)
0090            dlthta=zero
0091            print 303,theta,fmag,dlthta,dfdtht
0092          end if
0093          if(rpoly .eq. 1) then
0094      c     Test the magnitude of the f-function of the final angle
```

ITERAT

```
0095            fmag=cdabs(f)
0096            if(fmag .gt. fmag0) then
0097              print 304,fmag0,fmag
0098              theta=theta0
0099            end if
0100          end if
0101          if(typitr .gt. 0) then
0102            if(typitr .eq. 1) then
0103              dfdtht=(rbar22*r22-zone)*dfdtht
0104            else
0105              dfdtht=(rbar11*r11-zone)*dfdtht
0106            end if
0107          end if
0108          return
0109    300   format('0Iterations:    exact')
0110    301   format('0Iterations:  inexact')
0111    302   format(8x,'real    imag     f mag       d real     d imag',5x,
0112         $       'dfdt real  dfdt imag')
0113    303   format(5x,2f8.4,1pe12.3,2(1x,2e11.3))
0114    304   format(' Warning ITERAT:  During RPOLY=1, starting fmag (',
0115         $       1pe10.4,') is smaller than final fmag (',1pe10.4,')')
0116          end
```

```
0001            subroutine mdhnkl (z,h1,h2,h1prme,h2prme,theta,idbg)
0002      c
0003            implicit complex*16 (a-h,o-z)
0004            complex*16 i,mpower,mterm
0005            real*8 a,b,c,d,cap,part1,part2,zmag
0006            character*4 idbg
0007            dimension a(30), b(30), c(30), d(30), cap(30), part1(2), part2(2)
0008            equivalence (part1,term4), (part2,sum4)
0009            data a   /  9.3043671692922944819d-01,  3.1014557230974314911d+01,
0010           $              2.0676371487316209897d+02,  5.7434365242545027449d+02,
0011           $              8.7021765519007617234d+02,  8.2877871922864397320d+02,
0012           $              5.4168543740434246542d+02,  2.5794544638302022111d+02,
0013           $              9.3458495066311674231d+01,  2.6626351870744066662d+01,
0014           $              6.1210004300561072794d+00,  1.1592803844803233472d+00,
0015           $              1.8401275944132116616d-01,  2.4833030963741048003d-02,
0016           $              2.8842080097260218300d-03,  2.9133414239656786138d-04,
0017           $              2.5827494893312753646d-05,  2.0256858739853140063d-06,
0018           $              1.4155736366074870734d-07,  8.8695090013000443124d-09,
0019           $              5.0110220346327933889d-10,  2.5658074934115685526d-11,
0020           $              1.9618064960912286666d-12,  5.0988092481207283185d-14,
0021           $              1.9948392989517716388d-15,  7.1886100863126905797d-17,
0022           $              2.3938095525516785112d-18,  7.3883010881224645255d-20,
0023           $              2.1194208514407528762d-21,  5.6653858632471341093d-23/
0024            data b   /  6.7829872514427588456d-01,  1.1304978752404598033d+01,
0025           $              5.3833232154307609704d+01,  1.1962940478735024376d+02,
0026           $              1.5337103177865415841d+02,  1.2780919314887846509d+02,
0027           $              7.4742218215718400631d+01,  3.2355938621523117060d+01,
0028           $              1.0785312873841039006d+01,  2.8532573740320209005d+00,
0029           $              6.1360373635097223595d-01,  1.0937678009821251966d-01,
0030           $              1.6422939954686564465d-02,  2.1055051223957133911d-03,
0031           $              2.3316778764072130571d-04,  2.2528288660939256561d-05,
0032           $              1.9156708045016374595d-06,  1.4446989475879618839d-07,
0033           $              9.7286124416697769730d-09,  5.8854279743918795891d-10,
0034           $              3.2160808603234314644d-11,  1.5952782045255116351d-12,
0035           $              7.2151886229105003778d-14,  2.9876557444763976717d-15,
0036           $              1.1368553061173507104d-16,  3.9889659863766691603d-18,
0037           $              1.2946984700995355913d-19,  3.8985199340546088228d-21,
0038           $              1.0920223904914870636d-22,  2.8527230681595795812d-24/
0039            data c   /  4.6521835846461472410d-01,  6.2029114461948629822d+00,
0040           $              2.5845464359145262382d+01,  5.2213059311404570392d+01,
0041           $              6.2158403942148298012d+01,  4.8751689366390821897d+01,
0042           $              2.7084271870217123228d+01,  1.1215019407957400909d+01,
0043           $              3.5945575025504490022d+00,  9.1815006450841609147d-01,
0044           $              1.9128126343925335199d-01,  3.3122296699437809740d-02,
0045           $              4.8424410379295043444d-03,  6.0568368204246458321d-04,
0046           $              6.5550182039227768583d-05,  6.1985987743950608612d-06,
0047           $              5.1654989786625507119d-07,  3.8220488188402150986d-08,
0048           $              2.5278100653705126277d-09,  1.5033066103898380141d-10,
0049           $              8.0822936042464409157d-12,  3.9473961437101054471d-13,
0050           $              1.7590891906016512675d-14,  7.1814214762263778920d-16,
0051           $              2.6957287823672589641d-17,  9.3358572549515461865d-19,
0052           $              2.9922619406895981315d-20,  8.9015675760511620701d-22,
0053           $              2.4644428505125033375d-23,  6.3656020935361057409d-25/
0054            data d   /  6.7829872514427588456d-01,  4.5219915009618392131d+01,
0055           $              3.7683262508015326776d+02,  1.1962940478735024344d+03,
0056           $              1.9938234131225040548d+03,  2.0449470903820554375d+03,
0057           $              1.4201021460986496090d+03,  7.1183064967350857463d+02,
```

```
0058          $            2.6963282184602597492d+02,  7.9891206472890[illegible]+02,
0059          $            1.9021715826880139294d+01,  3.7188105233392256682d+00,
0060          $            6.0764877832340288572d-01,  8.4220204895828535644d-02,
0061          $            1.0026214868551016149d-02,  1.0363012784032058021d-03,
0062          $            9.3867869420580235442d-05,  7.5124345274574017960d-06,
0063          $            5.3507368429183773360d-07,  3.4135482251472901638d-08,
0064          $            1.9618093247972931935d-09,  1.0209780508963274472d-10,
0065          $            4.8341763773500352579d-12,  2.0913590211334783723d-13,
0066          $            8.2990437346566602039d-15,  3.0316141496462685641d-16,
0067          $            1.0228117913786331176d-17,  3.1967863459247792364d-19,
0068          $            9.2821903191776400453d-21,  2.5103962999804300309d-22/
0069           data cap /  1.0416666666666666663d-01,  8.3550347222222222116d-02,
0070          $            1.2822657455632716019d-01,  2.9184902646414046315d-01,
0071          $            8.8162726744375764874d-01,  3.3214082818627675264d+00,
0072          $            1.4995762986862554546d+01,  7.8923013011586517530d+01,
0073          $            4.7445153886826431887d+02,  3.2074900908906619004d+03,
0074          $            2.4086549640874004605d+04,  1.9892311916950979121d+05,
0075          $            1.7919020077753438063d+06,  1.7484377180034121023d+07,
0076          $            1.8370737967633072978d+08,  2.0679040329451551508d+09,
0077          $            2.4827519375935888472d+10,  3.1669454981734887315d+11,
0078          $            4.2771126865134715582d+12,  6.0971132411392560749d+13,
0079          $            9.1486942234356396792d+14,  1.4413525170009350101d+16,
0080          $            2.3788844395175757942d+17,  4.1046081600946921885d+18,
0081          $            7.3900049415704853993d+19,  1.3859220004603943141d+21,
0082          $            2.7030825930275761623d+22,  5.4747478619645573335d+23,
0083          $            1.1498937014386333524d+25,  2.5014180692753603969d+26/
0084           data i/(0.d0,1.d0)/
0085           data one/(1.d0,0.d0)/,two/(2.d0,0.d0)/,zero/(0.d0,0.d0)/
0086           data root3/(1.73205080756888d0,0.d0)/
0087           data alpha/(8.53667218838951d-1,0.d0)/
0088           data const1/( 2.58819045102522d-01,-9.65925826289067d-01)/
0089           data const2/( 2.58819045102522d-01, 9.65925826289067d-01)/
0090           data const3/(-9.65925826289067d-01, 2.58819045102522d-01)/
0091           data const4/(-9.65925826289067d-01,-2.58819045102522d-01)/
0092    c
0093           zpower=one
0094           sum3=zero
0095           sum4=zero
0096           zmag=cdabs(z)
0097           if(zmag .gt. 6.1d0) go to 70
0098           sum1=zero
0099           sum2=zero
0100           zterm=-z**3/(200.d0,0.d0)
0101           do 50 m=1,30
0102           sum1=sum1+dcmplx(a(m),0.d0)*zpower
0103           sum2=sum2+dcmplx(b(m),0.d0)*zpower
0104           sum3=sum3+dcmplx(c(m),0.d0)*zpower
0105           term4=dcmplx(d(m),0.d0)*zpower
0106           sum4=sum4+term4
0107           if(dabs(part1(1)) .le. 1.d-17*dabs(part2(1)) .and.
0108          $   dabs(part1(2)) .le. 1.d-17*dabs(part2(2))) go to 60
0109    50     zpower=zpower*zterm
0110    60     gm2f=i*(z*sum2-two*sum1)/root3
0111           gpmfp=i*(sum4+two*z*z*sum3)/root3
0112           h1=z*sum2+gm2f
0113           h2=h1-two*gm2f
0114           h1prme=sum4+gpmfp
```

```
0115          h2prme=h1prme-two*gpmfp
0116          go to 999
0117     70   mpower=one
0118          sum1=one
0119          sum2=one
0120          rtz=cdsqrt(z)
0121          sqrtzb=rtz*z
0122          zterm=i/sqrtzb
0123          mterm=-zterm
0124          dm=zero
0125          term3=one
0126          do 80 m=1,30
0127          zpower=zpower*zterm
0128          mpower=mpower*mterm
0129          dm=dm+one
0130          term1=dcmplx(cap(m),0.d0)*zpower
0131          term2=dcmplx(cap(m),0.d0)*mpower
0132          if(cdabs(term2/term3) .ge. 1.d0) go to 81
0133          sum1=sum1+term1
0134          sum2=sum2+term2
0135          sum3=sum3+dm*term1
0136          term4=dm*term2
0137          sum4=sum4+term4
0138          if(dabs(part1(1)/part2(1)) .le. 1.d-17 .and.
0139         $   dabs(part1(2)/part2(2)) .le. 1.d-17) go to 81
0140     80   term3=term2
0141     81   zterm=(-1.5d0,0.d0)/z
0142          sum3=sum3*zterm
0143          sum4=sum4*zterm
0144          term1=((-0.25d0,0.d0)-i*sqrtzb)/z
0145          term2=((-0.25d0,0.d0)+i*sqrtzb)/z
0146          exp1=cdexp((0.d0,0.66666666666666667d0)*sqrtzb)
0147          exp2=const1*exp1
0148          exp3=const2/exp1
0149          exp4=const3*exp1
0150          exp5=const4/exp1
0151          zterm=alpha/cdsqrt(rtz)
0152          term4=z
0153          if(part1(1) .ge. 0.d0 .or. part1(2) .ge. 0.d0) go to 90
0154          h1=zterm*(exp2*sum2+exp5*sum1)
0155          h1prme=zterm*(exp2*(sum2*term2+sum4)+exp5*(sum1*term1+sum3))
0156          go to 110
0157     90   h1=zterm*exp2*sum2
0158          h1prme=zterm*exp2*(sum2*term2+sum4)
0159     110  if(part1(1) .ge. 0.d0 .or. part1(2) .lt. 0.d0) go to 120
0160          h2=zterm*(exp3*sum1+exp4*sum2)
0161          h2prme=zterm*(exp3*(sum1*term1+sum3)+exp4*(sum2*term2+sum4))
0162          go to 999
0163     120  h2=zterm*exp3*sum1
0164          h2prme=zterm*exp3*(sum1*term1+sum3)
0165     c    calculate wronskian as partial check on validity
0166     999  sum4=h1*h2prme-h1prme*h2
0167          if(dabs(part2(1)) .le. 1.d-8 .and.
0168         $   dabs(part2(2)+1.457495441040461d0) .le. 1.d-8) go to 1000
0169          print 1001,sum4,theta,idbg
0170     1000 return
0171     1001 format(' ***** possible error in mdhnkl:  w = ',1p2e15.6,
```

```
0172          $       ' for theta = ',0p2f10.4,' at ',a4)
0173           end
```

```
0001            subroutine newmag(j,r,phij,thet,bmf,dip,b,br,bp,bt)
0002    c
0003    c       Returns parameters of the geomagnetic field
0004    c
0005    c       Input:  J=0:  Use spherical earth
0006    c               J=1:  Use spheroidal earth
0007    c               R     is altitude in km
0008    c               PHIJ is West longitude in radians
0009    c               THET is co-latitude in radians
0010    c
0011    c       Output:  BMF is declination of the geomagnetic field
0012    c                DIP is dip angle
0013    c                B   is total field
0014    c                BR  is radial component
0015    c                BP  is longitudinal component
0016    c                BT  id latitudinal component
0017    c
0018            dimension g(10,10), bm(10)
0019            data g/.0,3.032193e04,2.522093e03,-3.285459e03,-4.170639e03,1.6928
0020           $19e03,-6.684202e02,-1.900312e03,-2.405232e02,-9.358495e02,-5.75507
0021           $0e03,2.131549e03,-5.206994e03,6.237642e03,-4.496227e03,-3.650850e0
0022           $3,-1.241578e03,2.029996e03,-4.463745e02,-3.659410e02,3.495705e03,-
0023           $1.085898e02,-1.369823e03,-2.514676e03,-1.943789e03,-1.836598e03,-1
0024           $.313045e02,-1.626874e02,4.917246e02,-8.068787e02,1.220352e03,-4.75
0025           $3192e02,1.392784e02,-6.836385e02,8.297622e02,1.568303e02,2.302497e
0026           $03,-1.540896e02,5.700617e02,1.292881e03,-7.922399e02,1.080333e03,-
0027           $3.941087e01,2.055201e02,-1.853181e02,3.569555e02,-3.656370e01,3.01
0028           $2583e02,8.903696e01,-6.436587e02,-2.424140e02,-1.041800e03,5.89817
0029           $9e02,2.310479e02,-5.887414e01,4.001436e01,-1.209943e-02,9.459898e0
0030           $0,-1.050984e02,-3.745591e02,3.563806e02,-1.545264e03,-6.828717e02,
0031           $1.681499e02,2.971388e01,6.276772e00,7.309118e01,-3.402882e01,3.871
0032           $370e01,-1.670375e01,1.915876e03,7.079673e02,1.857451e02,-2.732077e
0033           $01,-1.705171e02,5.115862e01,1.302727e01,-3.776955e00,-2.940332e01,
0034           $3.510623e-01,-4.633602e02,6.821298e02,-2.394838e02,4.549622e02,-3.
0035           $794850e01,-1.617146e02,6.268821e00,1.004341e01,-4.002399e00,-4.152
0036           $194e00,2.803131e03,-1.698787e03,-4.244406e02,1.998351e02,6.192396e
0037           $01,-1.668931e02,-9.080082e01,-5.963821e-01,1.524572e00,-9.238670e-
0038           $01/
0039            data bm/9.933492e04,9.933492e04,3.746322e04,2.457753e04,1.329481e0
0040           $4,6.468820e03,3.385349e03,1.616258e03,7.409154e02,3.641040e02/
0041            data nmax/10/,berr/0.0001/
0042    c
0043    50      p22=abs(sin(thet))
0044            if(p22 .eq. 0.) p22=1.e-6
0045            p21=sqrt(1.-p22*p22)
0046            re=6356.912+p22*p22*(21.3677+.108*p22*p22)
0047            ar=(re+r)/6371.2
0048            if(thet .le. 1.570796327e0) go to 70
0049            p21=-p21
0050    70      if(j .eq. 0) go to 90
0051            ssq=p22*p22
0052            ar=ar+(14.288-ssq*(21.3677+.108*ssq))/6371.2
0053    90      ar=1./ar
0054    c       n= 2
0055            dp22=p21
0056    c       convert to east longitude
0057            phi=phij
```

```
0058            if(phi) 92,96,94
0059     92     phi=-phi
0060            go to 96
0061     94     phi=6.2831853e0-phi
0062     96     sp2=sin(phi)
0063            sp2=sin(phi)
0064            cp2=cos(phi)
0065            dp21=-p22
0066            aor=ar*ar*ar
0067            c2=g(2,2)*cp2+g(1,2)*sp2
0068            br=-(aor+aor)*(g(2,1)*p21+c2*p22)
0069            bt=aor*(g(2,1)*dp21+c2*dp22)
0070            bp=aor*(g(1,2)*cp2-g(2,2)*sp2)*p22
0071            if(nmax .lt. 3) go to 260
0072            aor=aor*ar
0073            err=berr*sqrt((bp/p22)**2+br**2+bt**2)
0074            if(bm(3)*aor .le. err) go to 260
0075            sp3=(sp2+sp2)*cp2
0076            cp3=(cp2+sp2)*(cp2-sp2)
0077            p31=p21*p21-0.333333333e0
0078            p32=p21*p22
0079            p33=p22*p22
0080            dp31=-p32-p32
0081            dp32=p21*p21-p33
0082            dp33=-dp31
0083            c2=g(3,2)*cp2+g(1,3)*sp2
0084            c3=g(3,3)*cp3+g(2,3)*sp3
0085            br=br-3.0*aor*(g(3,1)*p31+c2*p32+c3*p33)
0086            bt=bt+aor*(g(3,1)*dp31+c2*dp32+c3*dp33)
0087            bp=bp-aor*((g(3,2)*sp2-g(1,3)*cp2)*p32+2.0*(g(3,3)*sp3-g(2,3)*cp3)
0088           $*p33)
0089     c      n= 4
0090            if(nmax .lt. 4) go to 260
0091            aor=aor*ar
0092            if(bm(4)*aor .le. err) go to 260
0093            sp4=sp2*cp3+cp2*sp3
0094            cp4=cp2*cp3-sp2*sp3
0095            p41=p21*p31-0.26666666e0*p21
0096            dp41=p21*dp31+dp21*p31-0.26666666e0*dp21
0097            p42=p21*p32-0.20000000e0*p22
0098            dp42=p21*dp32+dp21*p32-0.20000000e0*dp22
0099            p43=p21*p33
0100            dp43=p21*dp33+dp21*p33
0101            p44=p22*p33
0102            dp44=3.0*p43
0103            c2=g(4,2)*cp2+g(1,4)*sp2
0104            c3=g(4,3)*cp3+g(2,4)*sp3
0105            c4=g(4,4)*cp4+g(3,4)*sp4
0106            br=br-4.0*aor*(g(4,1)*p41+c2*p42+c3*p43+c4*p44)
0107            bt=bt+aor*(g(4,1)*dp41+c2*dp42+c3*dp43+c4*dp44)
0108            bp=bp-aor*((g(4,2)*sp2-g(1,4)*cp2)*p42+2.0*(g(4,3)*sp3-g(2,4)*cp3)
0109           $*p43+3.0*(g(4,4)*sp4-g(3,4)*cp4)*p44)
0110            if(nmax .lt. 5) go to 260
0111            aor=aor*ar
0112            if(bm(5)*aor .le. err) go to 260
0113            sp5=(sp3+sp3)*cp3
0114            cp5=(cp3+sp3)*(cp3-sp3)
```

```
0115            p51=p21*p41-0.25714285e0*p31
0116            p52=p21*p42-0.22857142e0*p32
0117            dp51=p21*dp41+dp21*p41-0.25714285e0*dp31
0118            dp52=p21*dp42+dp21*p42-0.22857142e0*dp32
0119            p53=p21*p43-0.14285714e0*p33
0120            dp53=p21*dp43+dp21*p43-0.14285714e0*dp33
0121            p54=p21*p44
0122            dp54=p21*dp44+dp21*p44
0123            p55=p22*p44
0124            dp55=4.0*p54
0125            c2=g(5,2)*cp2+g(1,5)*sp2
0126            c3=g(5,3)*cp3+g(2,5)*sp3
0127            c4=g(5,4)*cp4+g(3,5)*sp4
0128            c5=g(5,5)*cp5+g(4,5)*sp5
0129            br=br-5.0*aor*(g(5,1)*p51+c2*p52+c3*p53+c4*p54+c5*p55)
0130            bt=bt+aor*(g(5,1)*dp51+c2*dp52+c3*dp53+c4*dp54+c5*dp55)
0131            bp=bp-aor*((g(5,2)*sp2-g(1,5)*cp2)*p52+2.0*(g(5,3)*sp3-g(2,5)*cp3)
0132           $*p53+3.0*(g(5,4)*sp4-g(3,5)*cp4)*p54+4.0*(g(5,5)*sp5-g(4,5)*cp5)*p
0133           $55)
0134      c     n= 6
0135            if(nmax .lt. 6) go to 260
0136            aor=aor*ar
0137            if(bm(6)*aor .le. err) go to 260
0138            sp6=sp2*cp5+cp2*sp5
0139            cp6=cp2*cp5-sp2*sp5
0140            p61=p21*p51-0.25396825e0*p41
0141            dp61=p21*dp51+dp21*p51-0.25396825e0*dp41
0142            p62=p21*p52-0.23809523e0*p42
0143            dp62=p21*dp52+dp21*p52-0.23809523e0*dp42
0144            p63=p21*p53-0.19047619e0*p43
0145            dp63=p21*dp53+dp21*p53-0.19047619e0*dp43
0146            p64=p21*p54-0.11111111e0*p44
0147            dp64=p21*dp54+dp21*p54-0.11111111e0*dp44
0148            p65=p21*p55
0149            dp65=p21*dp55+dp21*p55
0150            p66=p22*p55
0151            dp66=5.0*p65
0152            c2=g(6,2)*cp2+g(1,6)*sp2
0153            c3=g(6,3)*cp3+g(2,6)*sp3
0154            c4=g(6,4)*cp4+g(3,6)*sp4
0155            c5=g(6,5)*cp5+g(4,6)*sp5
0156            c6=g(6,6)*cp6+g(5,6)*sp6
0157            br=br-6.0*aor*(g(6,1)*p61+c2*p62+c3*p63+c4*p64+c5*p65+c6*p66)
0158            bt=bt+aor*(g(6,1)*dp61+c2*dp62+c3*dp63+c4*dp64+c5*dp65+c6*dp66)
0159            bp=bp-aor*((g(6,2)*sp2-g(1,6)*cp2)*p62+2.0*(g(6,3)*sp3-g(2,6)*cp3)
0160           $*p63+3.0*(g(6,4)*sp4-g(3,6)*cp4)*p64+4.0*(g(6,5)*sp5-g(4,6)*cp5)*p
0161           $65+5.0*(g(6,6)*sp6-g(5,6)*cp6)*p66)
0162            if(nmax .lt. 7) go to 260
0163            aor=aor*ar
0164            if(bm(7)*aor .le. err) go to 260
0165            sp7=(sp4+sp4)*cp4
0166            cp7=(cp4+sp4)*(cp4-sp4)
0167            p71=p21*p61-0.25252525e0*p51
0168            dp71=p21*dp61+dp21*p61-0.25252525e0*dp51
0169            p72=p21*p62-0.24242424e0*p52
0170            dp72=p21*dp62+dp21*p62-0.24242424e0*dp52
0171            p73=p21*p63-0.21212121e0*p53
```

```
0172          dp73=p21*dp63+dp21*p63-0.21212121e0*dp53
0173          p74=p21*p64-0.16161616e0*p54
0174          dp74=p21*dp64+dp21*p64-0.16161616e0*dp54
0175          p75=p21*p65-0.09090909e0*p55
0176          dp75=p21*dp65+dp21*p65-0.09090909e0*dp55
0177          p76=p21*p66
0178          dp76=p21*dp66+dp21*p66
0179          p77=p22*p66
0180          dp77=6.0*p76
0181          c2=g(7,2)*cp2+g(1,7)*sp2
0182          c3=g(7,3)*cp3+g(2,7)*sp3
0183          c4=g(7,4)*cp4+g(3,7)*sp4
0184          c5=g(7,5)*cp5+g(4,7)*sp5
0185          c6=g(7,6)*cp6+g(5,7)*sp6
0186          c7=g(7,7)*cp7+g(6,7)*sp7
0187          br=br-7.0*aor*(g(7,1)*p71+c2*p72+c3*p73+c4*p74+c5*p75+c6*p76+c7*p7
0188         $7)
0189          bt=bt+aor*(g(7,1)*dp71+c2*dp72+c3*dp73+c4*dp74+c5*dp75+c6*dp76+c7*
0190         $dp77)
0191          bp=bp-aor*((g(7,2)*sp2-g(1,7)*cp2)*p72+2.0*(g(7,3)*sp3-g(2,7)*cp3)
0192         $*p73+3.0*(g(7,4)*sp4-g(3,7)*cp4)*p74+4.0*(g(7,5)*sp5-g(4,7)*cp5)*p
0193         $75+5.0*(g(7,6)*sp6-g(5,7)*cp6)*p76+6.0*(g(7,7)*sp7-g(6,7)*cp7)*p77
0194         $)
0195    c     n= 8
0196          if(nmax .lt. 8) go to 260
0197          aor=aor*ar
0198          if(bm(8)*aor .le. err) go to 260
0199          sp8=sp2*cp7+cp2*sp7
0200          cp8=cp2*cp7-sp2*sp7
0201          p81=p21*p71-0.25174825e0*p61
0202          dp81=p21*dp71+dp21*p71-0.25174825e0*dp61
0203          p82=p21*p72-0.24475524e0*p62
0204          dp82=p21*dp72+dp21*p72-0.24475524e0*dp62
0205          p83=p21*p73-0.22377622e0*p63
0206          dp83=p21*dp73+dp21*p73-0.22377622e0*dp63
0207          p84=p21*p74-0.18881118e0*p64
0208          dp84=p21*dp74+dp21*p74-0.18881118e0*dp64
0209          p85=p21*p75-0.13986013e0*p65
0210          dp85=p21*dp75+dp21*p75-0.13986013e0*dp65
0211          p86=p21*p76-0.07692307e0*p66
0212          dp86=p21*dp76+dp21*p76-0.07692307e0*dp66
0213          p87=p21*p77
0214          dp87=p21*dp77+dp21*p77
0215          p88=p22*p77
0216          dp88=7.0*p87
0217          c2=g(8,2)*cp2+g(1,8)*sp2
0218          c3=g(8,3)*cp3+g(2,8)*sp3
0219          c4=g(8,4)*cp4+g(3,8)*sp4
0220          c5=g(8,5)*cp5+g(4,8)*sp5
0221          c6=g(8,6)*cp6+g(5,8)*sp6
0222          c7=g(8,7)*cp7+g(6,8)*sp7
0223          c8=g(8,8)*cp8+g(7,8)*sp8
0224          br=br-8.0*aor*(g(8,1)*p81+c2*p82+c3*p83+c4*p84+c5*p85+c6*p86+c7*p8
0225         17+c8*p88)
0226          bt=bt+aor*(g(8,1)*dp81+c2*dp82+c3*dp83+c4*dp84+c5*dp85+c6*dp86+c7*
0227         $dp87+c8*dp88)
0228          bp=bp-aor*((g(8,2)*sp2-g(1,8)*cp2)*p82+2.0*(g(8,3)*sp3-g(2,8)*cp3)
```

```
0229          $*p83+3.0*(g(8,4)*sp4-g(3,8)*cp4)*p84+4.0*(g(8,5)*sp5-g(4,8)*cp5)*p
0230          $85+5.0*(g(8,6)*sp6-g(5,8)*cp6)*p86+6.0*(g(8,7)*sp7-g(6,8)*cp7)*p87
0231          $+7.0*(g(8,8)*sp8-g(7,8)*cp8)*p88)
0232           if(nmax .lt. 9) go to 260
0233           aor=aor*ar
0234           if(bm(9)*aor .le. err) go to 260
0235           sp9=(sp5+sp5)*cp5
0236           cp9=(cp5+sp5)*(cp5-sp5)
0237           p91=p21*p81-0.25128205e0*p71
0238           dp91=p21*dp81+dp21*p81-0.25128205e0*dp71
0239           p92=p21*p82-0.24615384e0*p72
0240           dp92=p21*dp82+dp21*p82-0.24615384e0*dp72
0241           p93=p21*p83-0.23076923e0*p73
0242           dp93=p21*dp83+dp21*p83-0.23076923e0*dp73
0243           p94=p21*p84-0.20512820e0*p74
0244           dp94=p21*dp84+dp21*p84-0.20512820e0*dp74
0245           p95=p21*p85-0.16923076e0*p75
0246           dp95=p21*dp85+dp21*p85-0.16923076e0*dp75
0247           p96=p21*p86-0.12307692e0*p76
0248           dp96=p21*dp86+dp21*p86-0.12307692e0*dp76
0249           p97=p21*p87-0.06666666e0*p77
0250           dp97=p21*dp87+dp21*p87-0.06666666e0*dp77
0251           p98=p21*p88
0252           dp98=p21*dp88+dp21*p88
0253           p99=p22*p88
0254           dp99=8.0*p98
0255           c2=g(9,2)*cp2+g(1,9)*sp2
0256           c3=g(9,3)*cp3+g(2,9)*sp3
0257           c4=g(9,4)*cp4+g(3,9)*sp4
0258           c5=g(9,5)*cp5+g(4,9)*sp5
0259           c6=g(9,6)*cp6+g(5,9)*sp6
0260           c7=g(9,7)*cp7+g(6,9)*sp7
0261           c8=g(9,8)*cp8+g(7,9)*sp8
0262           c9=g(9,9)*cp9+g(8,9)*sp9
0263           br=br-9.0*aor*(g(9,1)*p91+c2*p92+c3*p93+c4*p94+c5*p95+c6*p96+c7*p9
0264          $7+c8*p98+c9*p99)
0265           bt=bt+aor*(g(9,1)*dp91+c2*dp92+c3*dp93+c4*dp94+c5*dp95+c6*dp96+c7*
0266          $dp97+c8*dp98+c9*dp99)
0267           bp=bp-aor*((g(9,2)*sp2-g(1,9)*cp2)*p92+2.0*(g(9,3)*sp3-g(2,9)*cp3)
0268          $*p93+3.0*(g(9,4)*sp4-g(3,9)*cp4)*p94+4.0*(g(9,5)*sp5-g(4,9)*cp5)*p
0269          $95+5.0*(g(9,6)*sp6-g(5,9)*cp6)*p96+6.0*(g(9,7)*sp7-g(6,9)*cp7)*p97
0270          $+7.0*(g(9,8)*sp8-g(7,9)*cp8)*p98+8.0*(g(9,9)*sp9-g(8,9)*cp9)*p99)
0271      c    n=10
0272           if(nmax .lt. 10) go to 260
0273           aor=aor*ar
0274           if(bm(10)*aor .le. err) go to 260
0275           sp10=sp2*cp9+cp2*sp9
0276           cp10=cp2*cp9-sp2*sp9
0277           p101=p21*p91-0.25098039e0*p81
0278           dp101=p21*dp91+dp21*p91-0.25098039e0*dp81
0279           p102=p21*p92-0.24705882e0*p82
0280           dp102=p21*dp92+dp21*p92-0.24705882e0*dp82
0281           p103=p21*p93-0.23529411e0*p83
0282           dp103=p21*dp93+dp21*p93-0.23529411e0*dp83
0283           p104=p21*p94-0.21568627e0*p84
0284           dp104=p21*dp94+dp21*p94-0.21568627e0*dp84
0285           p105=p21*p95-0.18823529e0*p85
```

```
0286            dp105=p21*dp95+dp21*p95-0.18823529e0*dp85
0287            p106=p21*p96-0.15294117e0*p86
0288            dp106=p21*dp96+dp21*p96-0.15294117e0*dp86
0289            p107=p21*p97-0.10980392e0*p87
0290            dp107=p21*dp97+dp21*p97-0.10980392e0*dp87
0291            p108=p21*p98-0.05882352e0*p88
0292            dp108=p21*dp98+dp21*p98-0.05882352e0*dp88
0293            p109=p21*p99
0294            dp109=p21*dp99+dp21*p99
0295            p1010=p22*p99
0296            dp1010=9.0*p109
0297            c2=g(10,2)*cp2+g(1,10)*sp2
0298            c3=g(10,3)*cp3+g(2,10)*sp3
0299            c4=g(10,4)*cp4+g(3,10)*sp4
0300            c5=g(10,5)*cp5+g(4,10)*sp5
0301            c6=g(10,6)*cp6+g(5,10)*sp6
0302            c7=g(10,7)*cp7+g(6,10)*sp7
0303            c8=g(10,8)*cp8+g(7,10)*sp8
0304            c9=g(10,9)*cp9+g(8,10)*sp9
0305            c10=g(10,10)*cp10+g(9,10)*sp10
0306            br=br-10.0*aor*(g(10,1)*p101+c2*p102+c3*p103+c4*p104+c5*p105+c6*p1
0307           $06+c7*p107+c8*p108+c9*p109+c10*p1010)
0308            bt=bt+aor*(g(10,1)*dp101+c2*dp102+c3*dp103+c4*dp104+c5*dp105+c6*dp
0309           1106+c7*dp107+c8*dp108+c9*dp109+c10*dp1010)
0310            bp=bp-aor*((g(10,2)*sp2-g(1,10)*cp2)*p102+2.0*(g(10,3)*sp3-g(2,10)
0311           $*cp3)*p103+3.0*(g(10,4)*sp4-g(3,10)*cp4)*p104+4.0*(g(10,5)*sp5-g(4
0312           $,10)*cp5)*p105+5.0*(g(10,6)*sp6-g(5,10)*cp6)*p106+6.0*(g(10,7)*sp7
0313           $-g(6,10)*cp7)*p107+7.0*(g(10,8)*sp8-g(7,10)*cp8)*p108+8.0*(g(10,9)
0314           $*sp9-g(8,10)*cp9)*p109+9.0*(g(10,10)*sp10-g(9,10)*cp10)*p1010)
0315    260     bp=bp/p22*1.e-5
0316            bt=bt*1.e-5
0317            br=br*1.e-5
0318            b=sqrt(br*br+bt*bt+bp*bp)
0319            bh=sqrt(bt*bt+bp*bp)
0320            bmf=3.141592654e0-acos(bt/bh)
0321            if(bp .lt. 0.) bmf=-bmf
0322            dip=acos(bh/b)
0323            if(br .gt. 0.) dip=-dip
0324            return
0325            end
```

```
0001            subroutine profin(lu,type,maxhts,nprint,nrspec,lmax,hlist,alogen)
0002      c
0003      c     Reads ionospheric profiles
0004      c     type=1:  electron and ion densities
0005      c          2:  collision frequencies
0006      c
0007            integer type
0008            character*80 bcd
0009            dimension hlist(maxhts),alogen(maxhts,3),en(3)
0010            if(type .ne. 2) then
0011              read(lu,1010) bcd
0012              if(nprint .gt. 1) print 1011,bcd
0013            end if
0014            do 202 l=1,maxhts+1
0015            read(lu,1020,end=900) ht,en
0016            if(ht .lt. 0.) then
0017              lmax=l-1
0018              return
0019            end if
0020            if(l .ne. 1 .and. ht .ge. hlist(l-1)) then
0021              print *,'ERROR PROFIN:  Profile heights out of order'
0022              go to 999
0023            end if
0024            hlist(l)=ht
0025            if(type .eq. 1 .and. nrspec .eq. 3) en(3)=en(2)-en(1)
0026              if(nprint .gt. 1) print 1021,ht,(en(k),k=1,nrspec)
0027            do 201 k=1,nrspec
0028      201   alogen(l,k)=alog(amax1(en(k),1.e-20))
0029      202   continue
0030            print *,'ERROR PROFIN:  Too many heights in profile'
0031            go to 999
0032      900   print *,'ERROR PROFIN:  Profile input not properly terminated'
0033      999   lmax=-1
0034            return
0035      1010  format(a80)
0036      1011  format(/1x,a80)
0037      1020  format(f7.2,4x,3e10.2)
0038      1021  format(f8.2,4x,1p3e10.2)
0039            end
```

```
0001          subroutine qartic(q,b3,b2,b1,b0,debug,newq)
0002          implicit real *8 (a-h,o-z)
0003          complex*16 b3,b2,b1,b0,q,b3sq,h,i,g,hprime,gprime,sqroot,
0004         $           p1,p2,cbert0,cbert1,cbert2,omega1,omega2,
0005         $           rootp,rootq,rootr,fncton,ctemp,dfdq,dq
0006          integer debug
0007          dimension diff(4),q(4)
0008          data omega1/(-5.d-1, 8.660254038d-1)/
0009          data omega2/(-5.d-1,-8.660254038d-1)/
0010          data tol/1.d-06/,imax/5/
0011     c
0012          iagain=0
0013          if(newq .eq. 1) go to 30
0014          newq=1
0015     10   b3sq=b3**2
0016          h=b2-b3sq
0017          i=b0-(4.d0,0.d0)*b3*b1+(3.d0,0.d0)*b2**2
0018          g=b1+b3*((-3.d0,0.d0)*b2+(2.d0,0.d0)*b3sq)
0019          hprime=-i/(12.d0,0.d0)
0020          gprime=-g**2/(4.d0,0.d0)-h*(h**2+(3.d0,0.d0)*hprime)
0021     c
0022          sqroot=cdsqrt(gprime**2+(4.d0,0.d0)*hprime**3)
0023          p1=(-.5d0,0.d0)*(gprime-sqroot)
0024          p2=(-.5d0,0.d0)*(gprime+sqroot)
0025          if(cdabs(p1) .lt. cdabs(p2)) p1=p2
0026          cbert0=cdexp(cdlog(p1)/(3.d0,0.d0))
0027          cbert1=omega1*cbert0
0028          cbert2=omega2*cbert0
0029     c
0030          rootp=cdsqrt(cbert0-hprime/cbert0-h)
0031          rootq=cdsqrt(cbert1-hprime/cbert1-h)
0032          rootr=cdsqrt(cbert2-hprime/cbert2-h)
0033          if(cdabs(g) .gt. 1.d-30) then
0034            sign=-rootp*rootq*rootr*(2.d0,0.d0)/g
0035            if(sign .lt. 0.d0) rootr=-rootr
0036          end if
0037          q(1)=+rootp+rootq+rootr-b3
0038          q(2)=+rootp-rootq-rootr-b3
0039          q(3)=-rootp+rootq-rootr-b3
0040          q(4)=-rootp-rootq+rootr-b3
0041     c
0042     30   if(debug .gt. 2) print 100,b3,b2,b1,b0
0043          do 60 n=1,4
0044          do 40 iter=1,imax
0045          fncton=(((q(n)+(4.d0,0.d0)*b3)*q(n)+(6.d0,0.d0)*b2)*q(n)
0046         $                +(4.d0,0.d0)*b1)*q(n)+b0
0047          dfdq=(((4.d0,0.d0)*q(n)+(12.d0,0.d0)*b3)*q(n)
0048         $                      +(12.d0,0.d0)*b2)*q(n)+(4.d0,0.d0)*b1
0049          dq=-fncton/dfdq
0050          q(n)=q(n)+dq
0051          testdq=cdabs(dq/q(n))
0052          if(testdq .le. tol) go to 60
0053     40   continue
0054          if(iagain .eq. 1) then
0055            fncton=(((q(n)+(4.d0,0.d0)*b3)*q(n)+(6.d0,0.d0)*b2)*q(n)
0056         $                 +(4.d0,0.d0)*b1)*q(n)+b0
0057            print 101,n,q(n),fncton,dq,iagain
```

```
0058            stop
0059          else
0060            iagain=1
0061            go to 10
0062          end if
0063    60    continue
0064    c
0065          l=0
0066          do 80 m=2,4
0067          do 80 n=m,4
0068          if(dimag(q(n)) .gt. 0.d0) go to 80
0069          l=l+1
0070          ctemp=q(n)
0071          q(n)=q(m-1)
0072          q(m-1)=ctemp
0073    80    continue
0074          if(l .eq. 2) go to 99
0075          do 81 n=1,4
0076          angq=cdang(q(n))*57.295779513d0
0077          if(angq .lt. 135.d0) angq=angq+360.d0
0078    81    diff(n)=dabs(angq-315.d0)
0079          do 82 nm=2,4
0080          do 82 n=nm,4
0081          if(diff(n) .gt. diff(nm-1)) go to 82
0082          temp=diff(n)
0083          diff(n)=diff(nm-1)
0084          diff(nm-1)=temp
0085          ctemp=q(n)
0086          q(n)=q(nm-1)
0087          q(nm-1)=ctemp
0088    82    continue
0089    c
0090    99    return
0091    100   format(/' In QARTIC:  b''s =',4(1pe13.4,e12.4))
0092    101   format(8h q root ,i1,2h =,1p2e13.5,3x,10hfunction =,2e13.5,3x,
0093         $       4hdq =,2e13.5,3x,8hiagain =,i1)
0094          end
```

```
0001          subroutine rbars
0002          implicit real *8 (a-h,o-z)
0003    c
0004          include 'common1.for'
0020          include 'common2.for'
0034          include 'common3.for'
0059    c
0060          complex*16 ngsq,sqroot,ratio,ikc,exd,exdsq,z1,z2,
0061         $           p0,h10,h20,h1prm0,h2prm0,caph10,caph20,
0062         $           pd,h1d,h2d,h1prmd,h2prmd,caph1d,caph2d,
0063         $           a1st,a2nd,a3rd,a4th,a1,a2,a3,a4,f1,f2
0064          real*8 kvraot,kvratt,ndsq,n0sq
0065    c
0066          ngsq=dcmplx(dble(epsr),-dble(sigma)/(omega*8.85434d-12))
0067          sqroot=cdsqrt(ngsq-ssq)
0068    c
0069          if(dimag(theta) .lt. -10.d0 .or. alpha .eq. 0.) go to 20
0070          if(d .eq. 0.) go to 10
0071    c
0072          kvraot=dexp(dlog(wn/alpha)/3.d0)
0073          kvratt=kvraot**2
0074          avrkot=1.d0/kvraot
0075          avrktt=avrkot**2*0.5d0
0076          n0sq=1.-alpha*h
0077          ratio=n0sq/ngsq*sqroot
0078          p0=kvratt*(n0sq-ssq)
0079          call mdhnkl(p0,h10,h20,h1prm0,h2prm0,theta,'rb 1')
0080          caph10=h1prm0*avrktt*h10
0081          caph20=h2prm0*avrktt*h20
0082          a1st=caph20-zmplxi*ratio*kvraot*h20
0083          a2nd=caph10-zmplxi*ratio*kvraot*h10
0084          a3rd=h2prm0-zmplxi*kvraot*sqroot*h20
0085          a4th=h1prm0-zmplxi*kvraot*sqroot*h10
0086          ndsq=1.-alpha*(h-d)
0087          pd=kvratt*(ndsq-ssq)
0088          call mdhnkl(pd,h1d,h2d,h1prmd,h2prmd,theta,'rb 2')
0089          caph1d=h1prmd*avrktt*h1d
0090          caph2d=h2prmd*avrktt*h2d
0091          f1=h2d*a2nd-h1d*a1st
0092          f2=h2d*a4th-h1d*a3rd
0093          a1=c*ndsq*f1
0094          a2=zmplxi*avrkot*(caph1d*a1st-caph2d*a2nd)
0095          a3=zmplxi*avrkot*(h2prmd*a4th-h1prmd*a3rd)
0096          a4=c*f2
0097          rbar11=(a1-a2)/(a1+a2)
0098          rbar22=(a3+a4)/(a4-a3)
0099          hg=exp(-.5*alpha*d)*(h20*a2nd-h10*a1st)/f1
0100          norm11=f1*f1
0101          norm22=f2*f2
0102          norm12=f1*f2
0103          return
0104    c
0105    10    rbar11=(ngsq*c-sqroot)/(ngsq*c+sqroot)
0106          rbar22=(c-sqroot)/(c+sqroot)
0107          hg=zone
0108          norm11=(-2.124292958d0,0.d0)
0109          norm22=norm11
```

```
0110            norm12=norm11
0111            return
0112      c
0113      c     flat earth
0114      20    ikc=dcmplx(0.d0,-wn)*c
0115            exd=cdexp(ikc*d)
0116            exdsq=exd*exd
0117            z1=(ngsq*c-sqroot)/(ngsq*c+sqroot)
0118            z2=(c-sqroot)/(c+sqroot)
0119            rbar11=z1*exdsq
0120            rbar22=z2*exdsq
0121            hg=exd*(zone+z1)/(zone+rbar11)
0122            norm11=(zone+rbar11)*(zone+rbar11)/exdsq
0123            norm22=(zone+rbar22)*(zone+rbar22)/exdsq
0124            norm12=(zone+rbar11)*(zone+rbar22)/exdsq
0125            return
0126            end
```

```
0001            subroutine recvr(tlng,tclt,xtr,rho,rlng,rclt)
0002    c
0003    c       Returns coordinates of a point which is at a specified great
0004    c       circle distance and bearing angle from the input point
0005    c
0006    c       Input:  TLNG is longitude of transmitter
0007    c               TCLT is co-latitude of transmitter
0008    c               XTR  is geographic bearing angle of receiver
0009    c               RHO  is great circle distance to the receiver
0010    c
0011    c       Output:  RLNG is longitude of receiver
0012    c                RCLT is co-latitude of receiver
0013    c
0014    c       All coordinates, RHO and XTR are in radians
0015    c       Sign convention is + for West and North
0016    c
0017            data pi/3.14159265e0/,twopi/6.28318531e0/
0018    c
0019            reduce(arg)=sign(amin1(abs(arg),1.),arg)
0020    c
0021            ctclt=cos(tclt)
0022            stclt=sin(tclt)
0023            br=xtr
0024            gcd=rho
0025    c
0026            if(abs(br) .lt. twopi) go to 2
0027            br=amod(br,twopi)
0028    2       if(br .ge. 0.) go to 3
0029            br=br+twopi
0030    3       if(gcd .lt. pi) go to 5
0031            gcd=twopi-gcd
0032            br=br+pi
0033            if(br .ge. twopi) br=br-twopi
0034    5       if(br .le. 1.e-6) go to 10
0035            if(abs(br-pi) .le. 1.e-6) go to 14
0036            if(abs(gcd-pi) .le. 1.e-6) go to 14
0037            cgcd=cos(gcd)
0038            sgcd=sin(gcd)
0039            crclt=ctclt*cgcd+stclt*sgcd*cos(br)
0040            srclt=sqrt(1.-crclt**2)
0041            rclt=acos(reduce(crclt))
0042            delta=acos(reduce((cgcd-ctclt*crclt)/(stclt*srclt)))
0043            if(br .lt. pi) delta=-delta
0044            rlng=tlng+delta
0045            go to 20
0046    c
0047    c       receiver is due north, south or on opposite longitude
0048    10      rclt=tclt-gcd
0049            if(rclt .lt. 0.) go to 12
0050    11      rlng=tl g
0051            crclt=cos(rclt)
0052            srclt=sin(rclt)
0053            go to 99
0054    12      rclt=-rclt
0055    13      rlng=tlng+pi
0056            crclt=cos(rclt)
0057            srclt=sin(rclt)
```

```
0058              go to 20
0059     14       rclt=tclt+gcd
0060              if(rclt .lt. pi) go to 11
0061              rclt=twopi-rclt
0062              go to 13
0063     c
0064     20       if(rlng .gt.  pi) go to 21
0065              if(rlng .lt. -pi) go to 22
0066              go to 99
0067     21       rlng=rlng-twopi
0068              go to 99
0069     22       rlng=rlng+twopi
0070     c
0071     99       return
0072              end
```

```
0001          subroutine rplynm
0002          implicit real *8 (a-h,o-z)
0003      c
0004          include 'common2.for'
0018          include 'common3.for'
0043      c
0044          complex*16 lgmtrx(30,4),prod,tlist1,tlist2
0045          complex* 8 stheta
0046          real*4 dst(30)
0047          integer use(30)
0048      c
0049          m=1
0050      10  if(m .le. 30 .and. tlist(1,m) .gt. 0 ) then
0051            thetar=tlist(1,m)
0052            theta=tlist(2,m)
0053            c=cdcos(theta*zdtr)
0054            csq=c*c
0055            s=cdsin(theta*zdtr)
0056            ssq=s*s
0057            call  nteg
0058            do 12 n=1,4
0059      12      lgmtrx(m,n)=logrs(n)
0060            adjflg=1
0061            m=m+1
0062            go to 10
0063          end if
0064          max=m-1
0065          if(max .le. 1) then
0066            print *,'ERROR RPLYNM   Insufficient tlist'
0067            stop
0068          else
0069            jmax=min0(max,nrtlst)
0070            adjflg=0
0071            return
0072          end if
0073      c
0074          entry rspoly
0075      c   Distance from theta to test angles
0076          stheta=theta
0077          do 24 i=1 max
0078          use(i)=i
0079      24  dst(i)=sqrt((real(stheta)-tlist(1,i))**2+
0080         $            (aimag(stheta)-tlist(2,i))**2)
0081      c   Order test angles according to distance
0082          call sortr(dst,max,use,max,1,max)
0083      c   Use only nrtlst angles
0084          do 50 n=1,4
0085           logrs(n)=0
0086          do 45 i=1,jmax
0087           i=use(i)
0088          tlist1=dcmplx(dble(tlist(1,i)),dble(tlist(2,i)))
0089          prod=zone
0090          do 44 j2=1,jmax
0091           j2=use(j2)
0092           if(i .ne. j2) then
0093            tlist2=dcmplx(dble(tlist(1,j2)),dble(tlist(2,j2)))
0094            prod=prod*(theta-tlist2)/(tlist1-tlist2)
```

RPLYNM

```
0095            end if
0096     44     continue
0097     45     logrs(n)=logrs(n)+prod*lgmtrx(i1,n)
0098     50     rs(n)=cdexp(logrs(n))
0099            return
0100            end
```

```
0001          subroutine savemc
0002     c
0003     c    This routine writes the mode parameters out to the logical unit
0004     c    defined by LUNIT7.
0005     c
0006          include 'common1.for'
0022          include 'common2.for'
0036     c
0037          write(lunit7,100) rho,freq,azim,codip,magfld,sigma,epsr,hprout
0038          do 10 m=1,modes
0039     10   write(lunit7,101) tp(m),nterm(m),t term(1,m),t term(2,m),
0040         $                  tp(m),nterm(m),t term(3,m),t term(4,m)
0041          write(lunit7,102)
0042          return
0043     c
0044     100  format('r',f7.3,' f',f8.4,' a',f8.3,' c',f8.3,' m',e10.3,
0045         $       ' s',1pe10.3,' e',0pf5.1,' t',f5.1)
0046     101  format('1',0p2f9.5,i1,1p4e15.8/'2',0p2f9.5,i1,1p4e15.8)
0047     102  format(' ')
0048          end
```

```
0001            subroutine sortr(array,nra,index,nri,ii,jj)
0002      c
0003      c     algorithm 347,r.c.singleton,communications of the acm,v12,n3,mar69
0004      c     sorts array into order of increasing value, from index ii to jj
0005      c     also orders index simultaneously if nri gt 1
0006      c     the only arithmetic operation on array is subtraction
0007      c     the user should consider the possibility of integer overflow
0008      c     arrays iu(k) and il(k) permit sorting up to 2**(k+1)-1 elements
0009      c
0010            dimension array(1),index(1),iu(36),il(36)
0011            if(jj .gt. nra) print *,'warning from sortr:  jj gt nra'
0012            m=1
0013            i=ii
0014            j=jj
0015      5     if(i .ge. j) go to 70
0016      10    k=i
0017            ij=(i+j)/2
0018            t=array(ij)
0019            if(nri .le. 1) go to 15
0020            n=index(ij)
0021      15    if(array(i) .le. t) go to 20
0022            array(ij)=array(i)
0023            array(i)=t
0024            t=array(ij)
0025            if(nri .le. 1) go to 20
0026            index(ij)=index(i)
0027            index(i)=n
0028            n=index(ij)
0029      20    l=j
0030            if(array(j) .ge. t) go to 40
0031            array(ij)=array(j)
0032            array(j)=t
0033            t=array(ij)
0034            if(nri .le. 1) go to 25
0035            index(ij)=index(j)
0036            index(j)=n
0037            n=index(ij)
0038      25    if(array(i) .le. t) go to 40
0039            array(ij)=array(i)
0040            array(i)=t
0041            t=array(ij)
0042            if(nri .le. 1) go to 40
0043            index(ij)=index(i)
0044            index(i)=n
0045            n=index(ij)
0046            go to 40
0047      30    array(l)=array(k)
0048            array(k)=tt
0049            if(nri .le. 1) go to 40
0050            index(l)=index(k)
0051            index(k)=nn
0052      40    l=l-1
0053            if(array(l) .gt. t) go to 40
0054            tt=array(l)
0055            if(nri .le. 1) go to 50
0056            nn=index(l)
0057      50    k=k+1
```

SORTR

```
0058            if(array(k) .lt. t) go to 50
0059            if(k .le. l) go to 30
0060            if(l-i .le. j-k) go to 60
0061            il(m)=i
0062            iu(m)=l
0063            i=k
0064            m=m+1
0065            go to 80
0066    60      il(m)=k
0067            iu(m)=j
0068            j=l
0069            m=m+1
0070            go to 80
0071    70      m=m-1
0072            if(m .eq. 0) return
0073            i=il(m)
0074            j=iu(m)
0075    80      if(j-i .ge. 11) go to 10
0076            if(i .eq. ii) go to 5
0077            i=i-1
0078    90      i=i+1
0079            if(i .eq. j) go to 70
0080            t=array(i+1)
0081            if(nri .le. 1) go to 95
0082            n=index(i+1)
0083    95      if(array(i) .le. t) go to 90
0084            k=i
0085    100     array(k+1)=array(k)
0086            if(nri .le. 1) go to 105
0087            index(k+1)=index(k)
0088    105     k=k-1
0089            if(t .lt. array(k)) go to 100
0090            array(k+1)=t
0091            if(nri .le. 1) go to 90
0092            index(k+1)=n
0093            go to 90
0094            end
```

```
0001          subroutine wvguid
0002      c
0003      c     This routine drives the generation of mode parameters using the
0004      c     input elist.
0005      c     If RPOLY is 0, then all calculations are made exactly.
0006      c     If RPOLY is 2, then all calculations are made approximately using
0007      c     the routine RPLYNM.
0008      c     If RPOLY is 1, then the initial calculations are approximate to
0009      c     refine the initial solutions and the final solutions are obtained
0010      c     using the exact formulation.
0011      c
0012          include 'common1.for/list'
0013    1 c
0014    1     common/input/freq,rho,azim,codip,magfld,sigma,epsr,beta,hprime,
0015    1    $       hprout
0016    1     common/path/pathid,tlong,tlat,rlong,rlat,rbear,dmax,drmin,drmax,
0017    1    $       year,month,day,gmt,nprint,nprof,npath,igcd,ignd,mdir,lost,
0018    1    $       lunit7,lx
0019    1     common/ionosp/htlist(50),lnlist(50,3),hclist(50),cflist(50,3),
0020    1    $       charge(3),mratio(3),nrspec,lhtmx,lhtmn,lht,mhtmx,mhtmn,mht
0021    1 c
0022    1     character*80 pathid
0023    1     integer year,day
0024    1     real*4 freq,rho,azim,codip,magfld,sigma,epsr,beta,hprime,hprout,
0025    1    $       tlong,tlat,rlong,rlat,rbear,dmax,drmin,drmax,gmt,
0026    1    $       htlist,lnlist,hclist,cflist,charge,mratio
0027    1 c
0028          include 'common2.for/list'
0029    1 c
0030    1     common/wg_in/elist(2,30),tlist(2,30),dtheta(2),lub(2),deigen(2),
0031    1    $       thtinc,ftol,maxitr,alpha,h,d,prec,wr0,atnmax,debug,typitr,
0032    1    $       rpoly,nrtlst
0033    1     common/wg_out/tp(30),tterm(4,30),nterm(30),mode(30),modes,nmds
0034    1 c
0035    1     complex*8 tp,tterm,dthta
0036    1     integer debug,typitr,rpoly
0037    1     real*4 elist,tlist,dtheta,lub,deigen,thtinc,ftol,alpha,h,d,prec,
0038    1    $       wr0,atnmax
0039    1 c
0040    1     equivalence (dtheta,dthta)
0041    1 c
0042          include 'common3.for/list'
0043    1 c
0044    1     common/f_fnctn/omega,wn,thetar,thetai,c,s,csq,ssq,f,dfdtht,
0045    1    $       hg,norm11,norm22,norm12,rbar11,rbar22,
0046    1    $       nriter,newq,adjflg,isotrp
0047    1     common/r_matrx/r11,r22,r12,r21,
0048    1    $       logr11,logr22,logr12,logr21,
0049    1    $       dl11dh,dl22dh,dl12dh,dl21dh,ht,delh,topht
0050    1     common/m_matrx/m11,m12,m13,m21,m22,m23,m31,m32,m33
0051    1 c
0052    1     integer adjflg
0053    1     real*8 omega,wn,thetar,thetai,ht,delh,topht,r(8),logr(8),dlrdh(8)
0054    1     complex*16 theta,c,s,csq,ssq,f,dfdtht,
0055    1    $           hg,norm11,norm22,norm12,rbar11,rbar22,
0056    1    $           r11,r22,r12,r21,rs(4),
0057    1    $           logr11,logr22,logr12,logr21,logrs(4),
```

```
0058  1      $           dl11dh,dl22dh,dl12dh,dl21dh,dlrsdh(4),
0059  1      $           m11,m12,m13,m21,m22,m23,m31,m32,m33,
0060  1      $           zero/(0.d0,0.d0)/,zone/(1.d0,0.d0)/,
0061  1      $           zmplxi/(0.d0,1.d0)/,zdtr/(1.745329252d-2,0.d0)/
0062  1 c
0063  1       equivalence (thetar,theta),
0064  1      $            (rl1,rs),(logrl1,logrs),(dl11dh,dlrsdh),
0065  1      $            (rl1,r), (logrl1,logr), (dl11dh,dlrdh)
0066  1 c
0067    c
0068          complex*16 theta0,stp,ratio,store1,store2,store3,
0069         $           wterm,ecomp,mik
0070          complex* 8 eigen(30)
0071          real*8 cdang,reflht,capk,stpr,stpi
0072          integer psave
0073          character*20 reason,blank
0074          equivalence (elist,eigen)
0075          data blank/'                    '/,reflht/70.d0/
0076    c
0077          psave=rpoly
0078          capk=1 /(1.-.5*alpha*h)
0079          omega=6.283185306d3*freq
0080          wn=2.0958426d-2*freq
0081          wterm=dcmplx(0.d0,-.5d0*wn*reflht)
0082          mik=dcmplx(0.d0,-1.d3*wn)
0083          debug=nprint
0084          adjflg=0
0085          newq=0
0086          if(magfld .le. 1.e-10) then
0087            isotrp=1
0088          else
0089            if(codip eq 90. .and. (azim.eq.90. .or. azim.eq.270.)) then
0090              isotrp=2
0091            else
0092              isotrp=0
0093            end if
0094          end if
0095          call intcmp
0096          if(rpoly .eq. 1) call rplynm
0097          if(nprint .gt. 0) print 1010
0098    10    kn=0
0099          ms=0
0100          index=1
0101    13    if(elist(1,index) .eq. 0.) go to 62
0102          theta0=eigen(index)
0103          kn=kn+1
0104          ms=ms+1
0105          mn=mode(kn)
0106          reason=blank
0107    15    theta=theta0
0108          call iterat
0109          fmag=cdabs(f)
0110          if(nriter .ge. maxitr .and. fmag .gt. ftol) then
0111            write(reason,2000) fmag
0112            go to 50
0113          end if
0114          pmag=cdabs(rbar11*r12/(zone-rbar11*r11))
```

```
0115          thtr=thetar
0116          thti=thetai
0117          if(thti .ge. 0.) then
0118            write(reason,2001)
0119            go to 50
0120          end if
0121          if(kn .gt. 1) then
0122            do 30 kd=1,kn-1
0123            if(abs(thtr-elist(1,kd)) .gt. deigen(1)) go to 30
0124            if(abs(thti-elist(2,kd)) .gt. deigen(2)) go to 30
0125            write(reason,2002) kd
0126            go to 50
0127    30      continue
0128          end if
0129          eigen(kn)=theta
0130    33    if(ms .eq. mn) go to 35
0131          if(rpoly .eq. 0 .and. nprint .gt. 0) print 1003
0132          ms=ms+1
0133          go to 33
0134    35    if(rpoly .eq. 1) go to 60
0135    c
0136          if(nriter .gt. maxitr/2) then
0137            print *,'Warning WVGD:  Excessive iterations for this mode:'
0138            lost=2
0139          end if
0140          s=cdsin(theta*zdtr)
0141          stp=s*capk
0142          at=-8.6858896d3*wn*dimag(stp)
0143          vc=1.d0/dreal(stp)
0144          tp(mn)=-zmplxi*cdlog(cdsqrt(zone-stp*stp)+zmplxi*stp)/zdtr
0145    c
0146          ratio=cdsqrt(s)/(dfdtht/zdtr)
0147          store1=(zone+rbar11)**2*(zone-rbar22*r22)*ratio/rbar11
0148          store2=(zone+rbar11)*(zone+rbar22)*ratio
0149          store3=(zone+rbar22)**2*(zone-rbar11*r11)*ratio/rbar22
0150          ecomp=wterm*store1*(s*hg)**2
0151          wm=20.d0*dlog10(cdabs(ecomp))
0152          wa=cdang(ecomp)
0153          if(nprint .gt. 0) print 1011,theta0,mn,nriter,eigen(kn),fmag,
0154         $                                pmag,at,vc,wm,wa,tp(mn)
0155    c
0156          t term(1,mn)=store1/norm11
0157          t term(2,mn)=store3/norm22
0158          t term(3,mn)=store2/norm12*r21
0159          t term(4,mn)=r12/r21
0160          if(cdabs(zone-r11*rbar11) .ge. cdabs(zone-r22*rbar22)) then
0161            nterm(mn)=2
0162          else
0163            nterm(mn)=1
0164          end if
0165          go to 60
0166    c
0167    50    if(rpoly .eq. 1) go to 63
0168          if(nprint .gt. 0) print 1012,theta0,nriter,theta,fmag,pmag
0169          if(rho .eq. 0 .or. npath .eq. 1) then
0170            if(rpoly .eq. 1) go to 63
0171    c       OK to drop a mode at the transmitter
```

```
0172              if(kn .eq. 30 .or. index .eq. 30)  then
0173                kn=kn-1
0174                go to 62
0175              end if
0176              do 53 m=kn,30
0177     53       eigen(m)=eigen(m+1)
0178              eigen(30)=(0.,0.)
0179              go to 13
0180            else
0181              print *,'ERROR WVGD:  Lost mode',mn,' because ',reason
0182              lost=1
0183              go to 999
0184            end if
0185     c
0186     60     if(kn .eq. 30 .or. index .eq. 30) go to 62
0187            index=index+1
0188            go to 13
0189     62     nmds=kn
0190            eigen(nmds+1)=(0.,0.)
0191            if(nmds .eq. 0) go to 65
0192            if(rpoly .ne. 1) go to 999
0193     63     rpoly=0
0194            go to 10
0195     65     print *,'ERROR WVGD:  Lost all modes'
0196            lost=1
0197     c
0198     999    if(nprint .gt. 0) print 1003
0199            rpoly=psave
0200            return
0201     c
0202     1003   format(' ')
0203     1010   format(/5x,'initial',4x,'mode iter',6x,'eigen',8x,'mag f',5x,
0204           $       'mag p',5x,'atten',4x,'v/c',8x,'wait''s exc',8x,'theta''')
0205     1011   format(1x,2f7.3,i4,i5,2x,2f7.3,2(1x,1pe9.3),1x,0pf8.3,1x,f9.5,
0206           $       1x,f9.3,1x,f6.3,1x,2f8.3)
0207     1012   format(1x,2f7.3,4x,i5,2x,2f7.3,2(1x,1pe9.3))
0208     2000   format('fmag=',1pe8.2)
0209     2001   format('thetai .gt. 0.')
0210     2002   format('it matches mode',i3)
0211            end
```

DISTRIBUTION LIST

Assistant Secretary of Defense
(CMD, CONT, COMM & INTELL)
Attn: DASb (1)
The Pentagon, Rm 3E172
Washington, DC 20301-3040

Defense Communications Agency
Center for Command, Control & Communications
Attn: A320 (P. Bird)
Arlington Hall Station
Arlington, VA 22212-5409

Defense Communications Agency
Center for Command, Control & Communications
Attn: A730 (G. Jones)
Arlington Hall Station
Arlington, VA 22212-5409

Director
Defense Intelligence Agency
Attn: VP-TPO
Washington, DC 20340-6537

Director
Defense Intelligence Agency
Attn: DIR
Washington, DC 20340-6537

Director
Defense Intelligence Agency
Attn: RTS-2B
Washington, DC 20340-6537

Defense Technical Information Center
Cameron Station (12)
Alexandria, VA 22314-6145

Commander Livermore Det (FC-1)
Field Command/DNA
Lawrence Livermore National Lab
P.O. Box 808 (L-317)
Attn: FC-1
Livermore, CA 94550-0662

Director
Joint Strat TGT Planning Staff JCS
Attn: JLK, (Attn: DNA REP)
Offutt AFB
Omaha, NB 68113

Director
National Security Agency
Attn: B432 (C. Goedeke)
Ft. Meade, MD 20755-6000

Commander/Director
US Army Atmospheric Sciences Laboratory
Attn: SLCAS-AE-E
White Sands Missile Range, NM 88002

Commander
US Army Communications R&D Command
Attn: DRCO-COM-RY (W. Kesselman)
Fort Monmouth, NJ 07703

Commander
US Army Nuclear & Chemical Agency
Attn: Library
7500 Backlick Road
Building 2073
Springfield, VA 22150-3198

Defense Advanced Research Project Agency
Attn: Library
1400 Wilson Boulevard
Arlington, VA 22209-2308

Defense Communications Agency
Center for Command, Control & Communications
Attn: A200 (R. Crawford)
Arlington Hall Station
Arlington, VA 22212-5409

Defense Communications Engineering Center
Attn: 123 (Library)
1860 Wiehle
Reston, VA 22090-5285

Director
Defense Intelligence Agency
Attn: DT-1B
Washington, DC 20340-6537

Director
Defense Intelligence Agency
Attn: DC-7B
Washington, DC 20340-6537

Director
Defense Nuclear Agency
Attn: RAAE
Washington, DC 20305-1000

Commander
Field Command
Defense Nuclear Agency
Attn: FCTXE
Kirtland, NM 87115-5000

Joint Chiefs of Staff
Attn: C3S
Washington, DC 20301

Chief
Livermore Division Fld Command DNA
Lawrence Livermore Laboratory
Attn: FC-1
P.O. Box 808
Livermore, CA 94550-0662

Director
National Security Agency
Attn: Technical Library
Ft. Meade, MD 20755-6000

Commander
Harry Diamond Laboratories
Attn: SHCELD-NW-P
2800 Powder Mill Road
Adelphi, MD 20783-1197

Commander
US Foreign Science & Tech. Center
Attn: DRXST-SD
220 7th Street, NE
Charlottesville, VA 22901-5396

US Army Strategic Defense Command
Attn: ATC-O (W. Davies)
P.O. Box 1500
Huntsville, AL 35807

Commander
US Army Missile Command
Attn: DRSMI-YSO (J. Gamble)
Redstone Arsenal, AL 35898-5000

Chief of Naval Operations
Attn: OP 941D
Navy Department
Washington, DC 20350-2000

Chief of Naval Operations
Attn: OP-981N
Navy Department
Washington, DC 20350-2000

Commanding Officer
Naval Research Laboratory
Attn: Code 4180 (J. Goodman)
Washington, DC 20375-0001

Commander
Space and Naval Warfare Systems Command
Attn: PDE-110-11021 (G. Brunhart)
Washington, DC 20376-5002

Air Force Center for Studies & Analysis
Attn: AFCSA/SASC
Washington, DC 20330-5420

Commander
Rome Air Development Center, AFSC
Attn: LID (J. Rasmussen)
Hanscomb AFB, MA 01731

Lawrence Livermore National Laboratory
Attn: Tech. Info. Dept. Library
P.O. Box 808
Livermore, CA 94550-0662

Los Alamos National Scientific Laboratory
Attn: J. Wolcott
P.O. Box 1663
Los Alamos, NM 87545

Sandia National Laboratory
Attn: ORG 6440 (D. Dahlgren)
P.O. Box 5800
Albuquerque, NM 87185-5800

Sandia National Laboratory
Attn: ORG 3141 (Tech. Lib)
P.O. Box 5800
Albuquerque, NM 87185-5800

Aerospace Corporation
Attn: Irving M. Garfunkel
P.O. Box 92957
Los Angeles, CA 90009-2957

Boeing Company
Attn: M/S 8K-04 (D. Clauson)
P.O. Box 3707
Seattle, WA 98124-2207

General Electric Company
Space Systems Division
Attn: A. Steinmeyer
Valley Forge Space Center
P.O. Box 8555
Philadelphia, PA 19101-8555

Kaman Tempo
Attn: B. Gambill
P.O. Drawer QQ
Santa Barbara, CA 93102

Commander
US Army White Sands Missile Range
Attn: STEWS-TE-N (K. Cummings)
White Sands Missile Range, NM 88002

Chief of Naval Operations
Attn: OP-654
Navy Department
Washington, DC 20350-2000

Commanding Officer
Naval Intelligence Support Center
Attn: NISC 50
4301 Suitland Rd., Bldg. 5
Washington, DC 20390

Commanding Officer
Naval Underwater Systems Center
Attn: Code 3411 (J. Katan)
New London, CT 06320

Director
Strategic Systems Project Office
Attn: NSP-L63 (Tech Library)
Washignton, DC 20376-5002

AF Weapons Laboratory, AFSC
ATT: SUL
Kirtland AFB, NM 87117-6008

Commander in Chief
Strategic Air Command
Attn: NRJ/STINFO
Offutt AFB, NB 68113

Los Alamos National Scientific Laboratory
Attn: D. Sappenfield
P.O. Box 1663
Los Alamos, NM 87545

Los Alamos National Scientific Laboratory
Attn: J. Zinn
P.O. Box 1663
Los Alamos, NM 87545

Sandia National Laboratory
Attn: ORG 1231 (T.P. Wright)
P.O. Box 5800
Albuquerque, NM 87185-5800

U.S. Department of Commerce
NOAA/MASC Security Office (Attn: Wm.F Utlaut)
325 Broadway
Boulder, CO 80303

Analytical Systems Engineering Corporation
Attn: Radio Sciences
5 Old Concord Road
Burlington, MA 01803-5188

Computer Sciences Corporation
Attn: Mail Code 269 (A. Osborne)
6565 Arlington Boulevard
Falls Church, VA 22046

GTE Government Systems Corporation
Strategic Systems Division
Attn: A Murphy
1 Research Drive
Westborough, MA 01581

Kaman Tempo
Attn: DASIAC
P.O Drawer QQ
Santa Barbara, CA 93102

Kaman Tempo
Attn: R. Rutherford
P.O. Box 9202
Fountain Valley, CA 92728-9202

Johns Hopkins University
Applied Physics Laboratory
Attn: C. Meng
Johns Hopkins University
Laurel, MD 20707-6099

Lockheed Missiles & Space Co., Inc.
Attn: R. Sears
3251 Hanover Street
Palo Alto, CA 94304

Mission Research Corporation
Attn: Tech Library
P.O. Drawer 719
San Barbara, CA 93102 071

Mitre Corporation
Attn: J. Valiga
1820 Dolly Madison Blvd
McLean, VA 22102

Pacific Sierra Research Corp
Attn: E.C. Field, Jr.
12340 Santa Monica Boulevard
Los Angeles, CA 90025

R&D Associates
Attn: William J. Karzas
P.O. Box 9695
Marina Del Rey, CA 90295

R&D Associates
Attn: M.A. Ory
P O Box 9695
Marina Del Rey CA 90295

Rand Corporation
Attn Technical Library
P O Box 2138
Santa Monica CA 90406 2138

Science Applications International Corp
Attn Dr E McKay (GS 2)
1710 Goodridge Drive
McLean VA 22102

Stanford University
Radio Science Laboratory
Attn A Fraser Smith
Stanford CA 94305

Telecommunications Sciences
Attn R Buchner
13339 N Central Expressway
Suite 101
Dallas TX 75243

Johns Hopkins University
Applied Physics Laboratory
Attn: K. Potocki
Johns Hopkins Road
Laurel, MD 20702-6099

Lockheed Missiles Space Co., Inc.
Attn: J. Kumer
3215 Hanover Street
Palo Alto, CA 94304-1191

Massachusetts Institute of Technology
Lincoln Laboratory
Attn: D.M. Towle
P.O. Box 73
Lexington, MA 02173-0073

Mission Research Corporation
Attn: R. Kilb
P.O. Drawer 719
San Barbara, CA 93102-0719

Mitre Corporation
Attn: MS J104 (M.R. Drespl
Burlington Road
Bedford, MA 01730

R&D Associates
Attn: Forrest Gilmore
P O Box 9695
Marina Del Rey, CA 90295

R&D Associates
Attn Carl Greifinger
P.O. Box 9695
Marina Del Rey CA 90295

R&D Associates
Attn R P Turco
P O Box 9695
Marina Del Rey CA 90295

Rockwell International Corporation
Attn MS 460 340 (... Asterud
3200 E Renner Road
Richardson TX 75081

Stanford University
Attn R A Helliwell
Radio Science Laboratory
Stanford CA 94305

Stanford University
Radio Science Laboratory
Attn J. Katsufrakis
Stanford CA 94305

END

8-87

DTIC